A magnetometer instrument for Attitude Determination in a nanosatellite

Manuel Antonio Pettersen Lains

Department of Physics

University of Oslo

A thesis submitted for the degree of

Master of Science
Electronics and computer technology
(Elektronikk og datateknologi)

September 2011

This work I dedicate to
my son, for being a light at the end of the tunnel
my wife, for giving me strength and courage
my parents, for supporting me in everything I do
my brother, for opening my mind

Abstract

This thesis details the design and implementation of a Three-Axis Magnetometer (TAM) instrument. The main goal of this work is to test and get flight history for a magnetic sensor instrument that is suitable for the Attitude Determination and Control System of the CubeSTAR student satellite. The actual implementation of the instrument will be part of the Sounding Rocket Attitude Determination System (SRADS) of the ICI-3 rocket.

Different magnetic field sensor technologies were evaluated before a candidate sensor was chosen; the Honeywell HMC1043. Most of this thesis deals with analysis and implementation of this magnetometer. The implementation will be customized for a sounding rocket, while the core of the design takes a low voltage and low power profile, to emulate restrictions in a small solar-powered satellite. The work on this project spans the design, production, testing and gathering calibration data for the instrument, as well as integration to the ICI-3 sounding rocket. The rocket will be launched after the submission of this thesis, which means that data from the flight is not available to be presented at the time of writing. The conclusion of a working instrument is made on the basis of successful integration at Andøya Rocket Range (ARR).

Acknowledgements

First and foremost, I would like to thank Associate Professor Torfinn Lindem at the Electronics Group for his encouragement and guidance as my supervisor. His vast experience in the field of electronics and enthusiasm as the leader of the Electronics and Computer Technology study program at the University of Oslo (UiO) make it an honor to be his student.

The engineers at the Electronics Laboratory deserve my thanks. Stein-Lyng Nilsen, Halvor Strøm and Roar Danielsen have provided me with their professional expertise and the high tech lab equipment at their disposal. Without their help, this project would not have been possible to realize. I want to thank engineer Espen Trondsen at the Plasma and Space Physics Group for letting me benefit from his experience, and I also want to thank PhD Candidate Tore Andre Bekkeng for being a pillar of support in all aspects of this work. Engineer Steinar Skaug Nilsen at the Department of Physics Instrument Workshop has been very helpful in solving the mechanical challenges of this work.

A special mention is in place of Professor Jøran Moen at the Plasma and Space Physics Group for being a driving force of the Space Physics projects here at UiO.

Contents

Chapter 1

Background and motivation

This work was carried out as a cooperation between The Group for Plasma and Space Physics and the Electronics Group at the Institute of Physics at the University of Oslo. The main goal of this work is to test and get flight history for a magnetic sensor instrument suitable for the Attitude Determination and Control System of the CubeSTAR student satellite. The instrument designed in the course of this work is a Three-Axis Magnetometer, which will fly in the upcoming ICI-3 sounding rocket. In the coming sections I will describe the background behind these spacecraft projects and the role of the instrument I am to develop.

1.1 Space Weather

The term Space Weather is used when talking about changing environmental conditions in near-Earth space[1]. The interaction between the solar wind and the earth's magnetosphere gives rise to the Space Weather phenomenon. Massive explosions on the sun can result in harmful effects to satellites and astronauts outside the protective magnetosphere. An overview of Space Weather and its effect on satellite and long-range communication can be found in [8]. An effort is being made internationally to better understand the underlying mechanisms behind space weather, and how to better predict it. Turbulence and irregularities in the ionosphere can disturb satellite to ground signals, and the Total Electron Content (TEC) along the path of a GPS signal can introduce positioning errors leading to problems for e.g. ships travelling in the northern regions. For this reason, the space weather phenomenon is becoming increasingly important for Norway specifically, and the University of Oslo is involved in space weather research through several projects. I will discuss two of these projects in the coming sections.

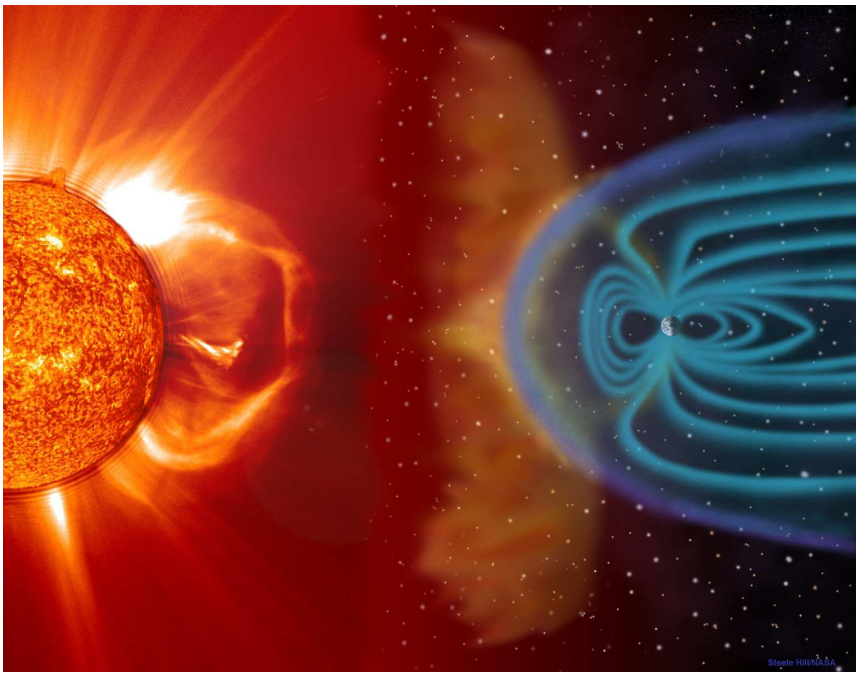

Figure 1.1: Illustration of the cause of space weather, from [31]. Earth's magnetic field creates a shielding bubble protecting the earth from bursts of particles and magnetic fields from the sun.

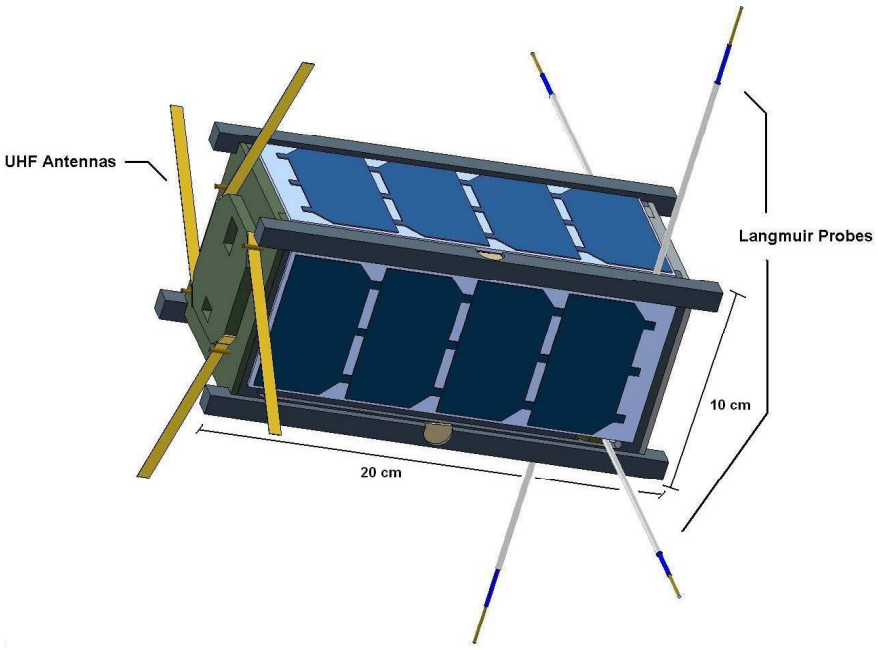

Figure 1.2: A model of the CubeSTAR satellite from [13]. The satellite will fly with the velocity vector pointing right and slightly downwards relative to the model. This way, the Langmuir Probes (see **Payload** in section 1.2.2) are at the front of the spacecraft and are subjected to as little turbulence as possible. The probes, as well as the UHF antennas at the rear, will spring out when the CubeSTAR is launched from the mother satellite.

1.2 CubeSTAR

CubeSTAR is a student satellite project, which main scientific goal is to measure electron density in the ionosphere [13]. Such data can be used to map structures in electron clouds, which is an integral part to understanding the space wheather phenomenon introduced in section 1.1. The main instrument on board is based on a novel multi-needle Langmuir Probe System developed at UiO [4]. With its four "needles" (illustrated in figure 1.2), it is able to measure absolute electron density with great spatial resolution. In addition to the scientific goal of the CubeSTAR, there is an academic goal to the project; to give students the ability to actively participate in the field of space science and technology through graduate theses, international conferences, and other related activities [13]. Therefore, students at UiO take part in building the satellite as part of their studies. In addition to the main instrument, there are a number of other subsystems, all developed by students at UiO. These are discussed in section 1.2.2. The satellite is planned for launch in 2013.

1.2.1 Weight and dimensions

The CubeSTAR is a nanosatellite based on the CubeSAT standard [29]. According to the standard, one CubeSAT unit (1U) measures $10cm * 10cm * 10cm$ with an allowed mass of up to $1.33kg$. The CubesSTAR satellite is a double unit (2U), meaning that the allowed dimensions and mass for the CubeSTAR are $20cm * 10cm * 10cm$ (see figure 1.2) with an allowed mass of up to $2.66kg$.

1.2.2 Subsystems

The CubeSTAR satellite consists of five subsystems. These are described briefly in the coming sections. The subsystems are listed below:

- Electronic Power System (EPS)
- Communication (COMM)
- On Board Data Handling (OBDH)
- Scientific Experiment (Payload)
- Attitude Determination and Control System (ADCS)

EPS

The EPS subsystem controls the conversion of solar energy to electrical power, the storing of energy in batteries as well as regulation of a continous and stable power supply for distribution throughout the satellite. The EPS of CubeSTAR should be able to provide a regulated voltage source of about 3V, with an estimated 2W of power available for the whole satellite [11].

[1]Meaning the region from and including the ionosphere and up to the interface between the earth's magnetosphere and the solar wind

COMM

The COMM subsystem is responsible for the satellite's communication with the ground, both for ground-based control of the satellite and transmission of collected measurement data back to earth. The system will operate in the amateur satellite band (UHF 435-438Mhz) [13], and will have both a high-speed link (up to 19200baud) and a low-speed link (up to 48 words/min).

OBDH

The On-Board Data Handling system is the main computer of the satellite. It dictates the flow of events, so to speak. It monitors the other subsystems of the satellite and is responsible for tasks such as resetting unresponsive subsystems. This system will be designed placing highest priority on integrity. It will implement a Triple Modular Redundancy (TMR) scheme, running three parallel processes, comparing them and performing a majority vote before taking any action.

Payload

The payload will consist of the scientific experiment; the multi-needle Langmuir Probe System. This will be a controller card interfacing the probes seen on figure 1.2. The measurements are dependent on the probes being spread out as far as possible and being at the front of the spacecraft during flight. This puts certain requirements on the ADCS subsystem.

ADCS

When the functionality of a spacecraft is dependent on its orientation, it is important to have an Attitude Determination and Control System (ADCS). In the case of the CubeSTAR satellite, optimal functionality of the main scientific instrument is dependent on the spacecraft travelling with the Langmuir probes at the front of the satellite (see figure 1.2). This is to avoid particle turbulence in the wake of the rest of the satellite structure. Consider the plane spanned by the probes and let us call it the "top wall" of the satellite. The normal vector pointing outwards from this wall (away from the center of the structure) should be aligned with the velocity vector. The maximum allowed deviation from this state is $\pm 10^\circ$ along the pitch and yaw axes [13]. The job of the ADCS is to make sure that the deviation is within these limits. This is done by employing various types of sensors (including a magnetometer) for determining the current orientation and spin of the satellite, as well as actuators (such as coils) for making adjustments to the current state. Attitude Determination is a motivation behind the instrument in this thesis and is discussed in further detail in chapter 2.

1.3 ICI-3

The ICI-3 is a sounding rocket which means that it is a rocket meant for taking measurements[2]. ICI stands for Investigation of Cusp Irregularities, which gives a pointer to the main mission of this series of rockets; to investigate the ionospheric cusp-regions and learn of the mechanisms making the plasma unstable. More detailed information on this can be found in [2]. The mission of the ICI-3 rocket is also connected to space weather exploration and its goal is largely related to the scientific goal of the CubeSTAR satellite.

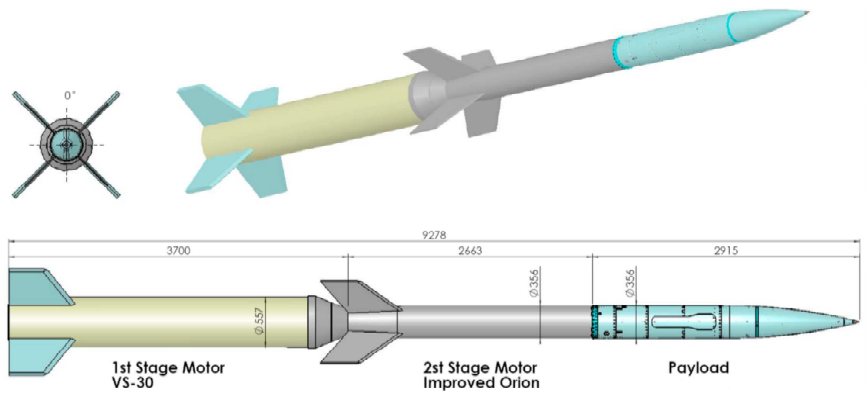

Figure 1.3: The ICI-3 sounding rocket [18] is a two stage ballistic rocket measuring almost 9.3 metres in length. The first stage motor is a Sonda VS-30, an Argentinian-Brazilian sounding rocket. It has a nominal thrust of 98,09 kN and a burning time of 19.17s. The second stage motor is an Improved Orion, an American sounding rocket, with a nominal thrust of 77.4kN and a burning time of 21.86s. At the top sits the payload structure housing all the electronics and instruments on board.

1.3.1 Structure and design

The ICI-3 rocket is about nine meters tall and weighs approximately 1.8 metric tons at ignition. As shown on figure 1.3 it consists of three main sections. The 1st Stage Motor, the 2nd Stage Motor, and the Payload section. The payload section has a length of about 2.9 meters and a diameter of 356mm, in which it houses all the electronics and measurement equipment. Figure 1.4 gives a closer look at the payload section.

[2]From nautical vocabulary, *to sound* has long been used about throwing a weighted line from a ship and in that way measure the water's depth. Nowadays, when used in the context of rockets, *sounding* means *taking a measurement*

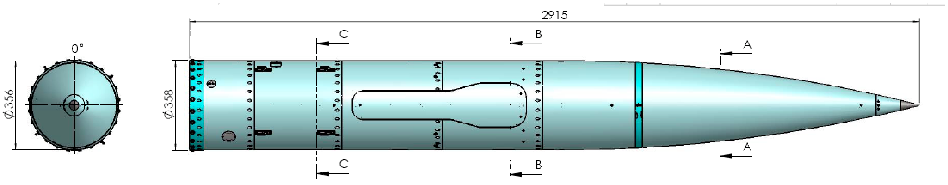

Figure 1.4: The ICI-3 payload. Section A is the nosecone section, housing Langmuir Probes and E-field booms as well as other electronic instruments. Section B is the electronic section which houses various electronic instruments, such as the SRADS Magnetometer instrument. Section C is the Hotel Section in which we find the aft booms for the AC/DC Magnetometer (ADM)

1.3.2 Instrumentation systems

The following systems are to be found on board the ICI-3:

- FBP : Fixed Bias Langmuir Probe, ISAS/JAXA
- m-NSLP : multi -Needle&Sphere Langmuir Probe system, UiO
- LEP-ESA : Low Energy Particle spectrometer, ISAS/JAXA
- EFW : Electric Field and Wave Experiment, UiO.
- ADM : AC/DC Magnetometer, LPP
- SRADS: Sounding Rocket Attitude Determination System, UiO

The magnetometer instrument developed during the work of this thesis is implemented as part of the SRADS system, so I will give a brief overview on that system in the next section. Details about the other on board instrumentation systems is beyond the scope of this thesis.

1.3.3 SRADS

The task of the Sounding Rocket Attitude Determination System (SRADS) is to determine the attitude (orientation) of the rocket at all times, so that measurement data from the scientific instruments on board can be related to a known attitude in a common reference frame (reference frames will be disussed in chapter 2). The SRADS system incorporates the following sensors:

- Three-Axis Magnetometer (TAM): The SRADS Magnetometer (SRADS MAG) is based on the Honeywell HMC1043. This is the instrument that is developed in the works of this thesis.
- Inertial Reference Unit (IRU): This unit incorporates three gyroscopes mounted orthogonally to eachother, for measuring roll, pitch and yaw rates respectively. The development of this unit is detailed in [1].

- Digital Sun Sensors: Two one-dimensional sun sensors are mounted on each side of the rocket (for redundancy) to measure the angle to the sun. As the rockets spins, they produce one sun angle measurement per rotation. The development of this unit is detailed in [12].
- Housekeeping Magnetometer: A backup commercial magnetometer.
- Housekeeping Accelerometer: A backup inertial sensor.

The functionality of the SRADS system as a whole is beyond the scope of this thesis and is detailed in [1]. General Attitude Determination theory, as well as the requirements of the SRADS MAG instrument, is discussed in chapter 2.

Chapter 2

Attitude determination

The Attitude Determination and Control System of a spacecraft has one goal; that is to make sure that the spacecraft is oriented the way we want it to be and has the spin or angular velocity we want it to have. To achieve that goal, the system employs various sensors to sense the rotation of the spacecraft as well as the absolute orientation. Having this information, it analyses the information and then employs actuators to make adjustments to the rotational movement of the spacecraft. Some common devices for performing these functions are introduced in the coming chapters.

2.1 Inertial sensors

Inertial sensors, as the name suggests, use their inherent inertia to detect a change in their positional or rotational state. As a result, they are not dependent on an external reference to give some sensible output. Two common types of inertial sensors used for Attitude Determination are accelerometers and gyroscopes.

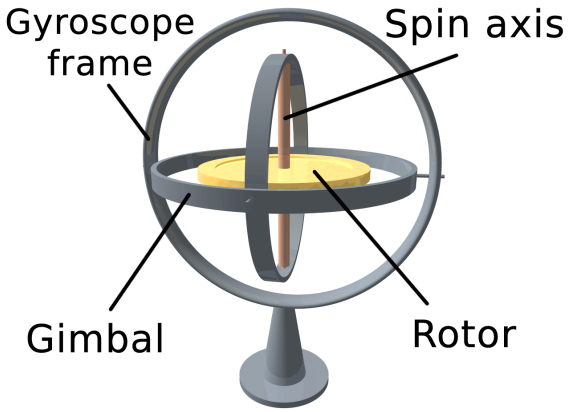

Figure 2.1: Illustration of a classical mechanical 3D Gyroscope based on two gimbals and a rotor. From a calibrated initial state, one can determine the new orientation based on the new state of the rotor and gimbals.[33]

2.1.1 Accelerometer

Accelerometers measure the spatial acceleration that the sensor experiences. For a spacecraft in free fall, the gravitational pull from the earth will be measured as zero, and so an accelerometer placed off the axis of rotation can be used to measure the centripetal acceleration it is exposed to, and from there derive the angular velocity [1]. Accelerometers nowadays can be made using MEMS[1] technology, which makes it possible to produce them small and cheap. A 3D accelerometer is normally built from three one-dimensional acceleration sensors.

2.1.2 Gyroscope

Gyroscopes detect angular rotation directly. The classical mechanical gyroscope uses a mechanical structure as seen on figure 2.1 to allow three degrees of freedom; pitch, roll and yaw. These gimballed units are very accurate, but mechanically complex, heavy and expensive, compared to the increasingly popular strapdown units [1]. Strapdown units are one-dimensional gyroscopes mounted at a fixed angle relative to the spacecraft. Three gyros are mounted orthogonally to each other, one for each axis of rotation. For high spin rates of up to thousands of degrees per second, it is more common to use rate gyros, which measure the rate of rotation directly. These can also be realized in MEMS technology, which makes for small and cheap units.

2.2 Attitude sensors

Attitude sensors detect the orientation of the spacecraft directly, by measuring some known external field or particles, thus finding out what orientation state the sensors find themselves in. There are several types of attitude sensors in use in spacecraft, among which we find star trackers, magnetic field sensors and sun sensors.

2.2.1 Star tracker

Star trackers use sensitive cameras to analyse the image of the sky. Knowing the approximate spacial coordinates of the spacecraft, this image can be related to a database of navigational stars and their positions and processed to derive the pointing direction of the camera relative to a known reference frame such as the Earth-Centered Inertial frame (see section 2.4.1). Star trackers excel in accuracy, as long as the spacecraft is stabilized. However, they are slow and require a somewhat stabilized vehicle to operate properly [14]. They can also be blinded by bright objects such as the sun or the moon, and are often used in conjunction with e.g. gyros for stabilization and as a backup during blinding phases. They are also physically large, complex and relatively expensive.

[1]MEMS: Micro Electro Mechanical System

2.2.2 Sun sensor

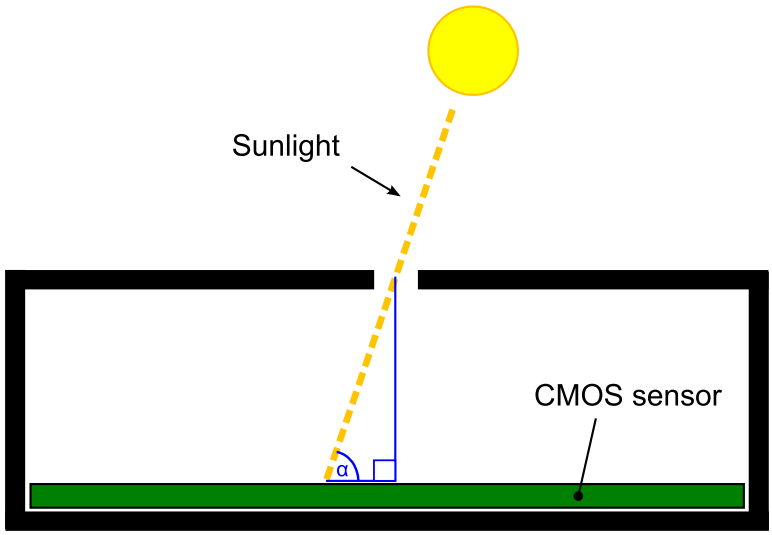

Figure 2.2: Principle of operation for a CMOS-based two axis sun sensor. The sunlight enters through a small hole and hits a spot on the internal CMOS chip. The chip is a two dimensional grid of pixels, and the lighted area determines the angle to the sun. In a one-dimensional simplification, we can get the angle α straight from the relation between height from the CMOS to the hole and the distance from the lighted area to the center of the chip.

Sun sensors detect sunlight and use it to determine the angle to the sun in Spacecraft-fixed Coordinates (see section 2.4.3). One way to accomplish this is to use a CMOS chip as shown in figure 2.2. The chip sits in a dark, closed compartment with only a small hole in the center to let light through. This way, the area of the chip detecting light (the calculated centre of intensity of the light) will indicate the angles to the sun in two dimensions. As this type of design will have less than 180° field of view (FOV), typically 130° [14] for each axis, multiple sensors must be used to cover the different sides of the spacecraft. One for each of the six sides of the spacecraft yields optimal results. In a spin-stabilized sounding rocket application, where a 3D vector to the sun is not the goal, a simplified setup can be used, where only one one-dimensional sensor is needed. For each rotation of the rocket a measurement of the sun angle will be taken. Development of such a system for the SRADS is detailed in [12].

2.2.3 Magnetometer

Magnetometers are used to detect magnetic fields. Tuned to the correct dynamic range, a 3-axis magnetic sensor can measure the 3D components of Earth's magnetic field, which constitute the vector of the magnetic field in Spacecraft-fixed Coordinates. Using a model such as the International Geomagnetic Reference Field (IGRF) and knowing the position of the spacecraft as well as the current date, the expected magnetic field vector seen from the spacecraft will be known at any given time. Some

vector rotations must be performed to relate the measured vector with the known vector, depending on how we want to use the data. This will be discussed in further detail in section 2.5.

2.3 Actuators

The word actuator comes from the verb *to actuate*, which means to put into mechanical action or motion. When it comes to spacecraft attitude control, actuators are the tools we can use to adjust the orientation the spacecraft. Two common types of actuators used in satellites are magnetic torquers and reaction wheels.

2.3.1 Magnetic Torquer

The magnetic torquer, or magnetorquer is essentially an electromagnetic coil. By running current through the coil a magnetic field will be created, which interacts with Earth's magnetic field. This causes a mechanical force called torque, or rotational momentum, which makes the objects rotate, just as when two magnets attract or repel eachother (e.g. compass needle). As the earth has a much larger mass than a spacecraft, the spacecraft will rotate and the earth will not (at least not by a measurable amount). Advantages to this type of actuator are that it contains no moving parts, and it is driven purely by electrical power. The disadvantage is that high magnetic flux densities are needed to create strong momentum.

2.3.2 Reaction Wheel

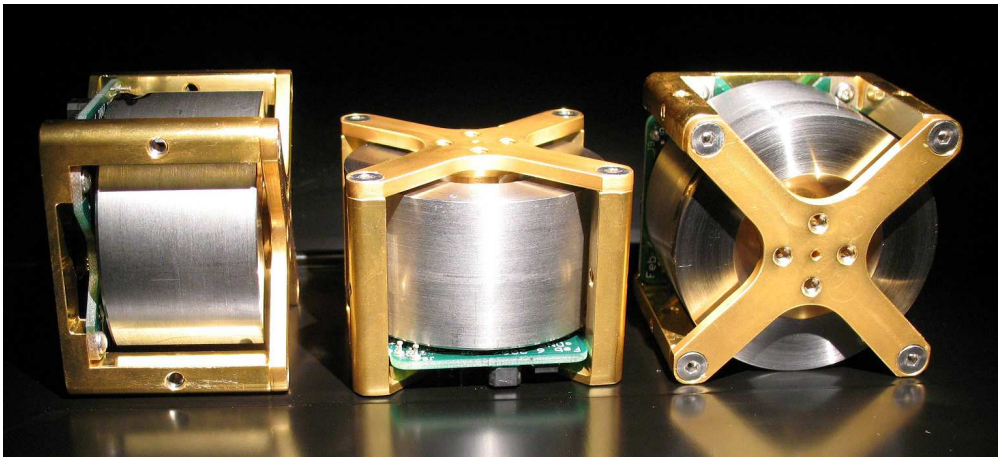

Figure 2.3: Reaction Wheels (from Sinclair Interplanetary). The mass of the wheels have enough inertia that rotating the wheels will make the whole spacecraft, to which they are fastened, rotate. To give full control over all three axes of rotation, three wheels must be employed.

Reaction wheels are essentially mechanical wheels connected to electromotors. They have enough mass so that when they accellerate their spin, they will apply a

rotational force to the whole spacecraft, making it rotate slightly. Figure 2.3 shows an example of reaction wheels that can be used for spacecraft attitude control. Reaction wheels can make very accurate adjustments to the attitude of a spacecraft, and are often used for aiming cameras or telescopes. During the course of operation, the wheels may build up significant momentum. For this reason they are often used together with other forms of actuators, such as magnetic torquers or propulsion systems, which can be used to dump momentum from the reaction wheels.

2.4 Reference Frames

Attitude determination in a spacecraft is the act of finding out which way the spacecraft is pointing relative to a known coordinate system. There are several common reference coordinate systems (so-called reference frames), each having their own specific usage area. Some of the most useful reference systems will be discussed in the coming sections.

2.4.1 Earth-centered Coordinate Systems

We will discuss two reference coordinate systems that are centered on earth, one is the Earth-Centered Earth Fixed (ECEF) reference frame and the other is the Earth-Centered Intertial (ECI). Both have their origin in the center of the earth, and both have their Z-axis pointing towards the Celestial Pole[2]. The difference between these frames is that while the X-axis of the ECEF points towards the intersection between the equator and the Greenwich Meridian, the X-axis of the ECI points towards the Vernal Equinox[3]. The Y-axis follows the Z- and X-axes.

2.4.2 Spacecraft Body Frame

This coordinate system is centered at the spacecraft itself, and rotates along with it. It is therefore the inherent coordinate system used on board the spacecraft. Sensors on board, like magnetometers and sun sensors, are aligned and calibrated according to this reference system, and measurements taken on board are given in this coordinate system. Figure 2.4 illustrates the axes of this frame of reference.

2.4.3 Spacecraft-centered Orbit Frame

The difference between the Orbit Frame coordinate system and the SBF is that this frame does not rotate with the spacecraft. It is fixed in relation to the orbit, so the X-axis points in the Nominal Velocity Vector direction, while the Z-axis points to the Nominal Nadir[4]. This coordinate system can be useful if we have in our database

[2]The Celestial Pole is the point through which the earth's axis of rotation points when followed upwards from the center out through the northern hemisphere

[3]The Vernal Equinox is the direction from the earth to the sun on the first day of spring when the sun crosses earth's equatorial plane

[4]Nadir being the direction from the center of the spacecraft towards the center of the planet.

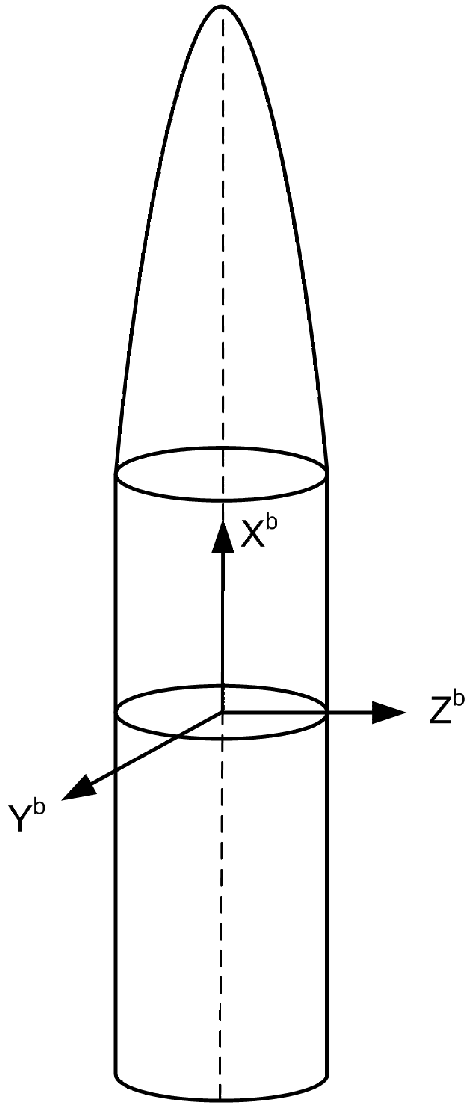

Figure 2.4: The axes of the Spacecraft Body Frame, from [1]. X^b points in the direction along the longitudinal axis of the spacecraft while the two other axes point orthogonally relative to that. The axes rotate with the aircraft.

the expected magnetic field vector to be measured at a specific point for the desired attitude. The difference between the vector we have in our database and the measured magnetic field vector will be the deviation from the nominal orbit orientation.

2.5 Attitude Determination with Magnetometer

Models such as IGRF can provide us with values for the strength and direction of the earth magnetic field for any geographical position and time. For a spacecraft such as the CubeSTAR, using this information we can build a database consisting of all the expected magnetic field values for each time quantum Δt of the satellite's expected flight. If the nominal (desired) flight orientation is that the SFB coordinate system is aligned with the Orbit Frame coordinate system, the magnetic vector values for each Δt can be calculated and stored as the magnetic field value in the Orbit Frame coordinate system. The magnetic field vector measured by the on-board magnetometer will be given in the SFC system, and the functionality of the ADCS will be to try to rotate the vehicle so that the measured magnetic field vector in the SFB system aligns with the nominal magnetic field vector in the Orbit Frame system.

2.6 CubeSTAR ADCS

As the CubeSTAR satellite is launched from the mother satellite, it is expected to be in a state of arbitrary spin. The first task of the ADCS will be to detumble the satellite. The detumbling phase consists of stopping the spin of the spacecraft, therefore any rate of change in the satellite orientation will be attempted nullified. This can be accomplished by using gyro measurements directly, or taking the derivative of the magnetic field measurements. Actuators will be used to apply opposite rotational force to the vehicle, making the spin stop. After the detumbling phase, the satellite will attempt to orient itself to the desired orientation. From there on, the task of the ADCS will be to adjust the orientation of the satellite as it moves along its orbit. In section 2.5 we discovered that the requirement of the magnetic field instrument in the Attitude Determination System of the CubeSTAR is to measure a magnetic field vector in the SBF coordinate system. It has been determined that the system will have a tolerance of a deviation of $\pm 10^\circ$ along the pitch and yaw axes. [13]. The actual calculations will be performed by a control system probably running a Kalman Filter, and is beyond the scope of this thesis.

2.7 ICI-3 SRADS

The difference between the ICI-3 SRADS system and the CubeSTAR system is that the ICI-3 does not have an Attitude Control System. It is therefore a possibility to process the data from the magnetometer after the actual flight without any realtime requirements. It does however require a fast sampling rate due to the fast spin of the rocket (nominally 4 rps, max 6 rps). The details of the desired properties of the

actual magnetometer implementation for the ICI-3 SRADS are discussed in section 3.4.

Chapter 3

Magnetic Sensor Technologies

There are various technologies available to measure a magnetic field vector in three dimensions. We will take a look at a few of the most prominent ones.

3.1 Hall Effect

This is the most commonly used technology for measuring magnetic fields, and has also been around for a long time. It is based on the Hall effect, discovered by Edwin Hall in 1879. A brief description of the measurement principle follows.

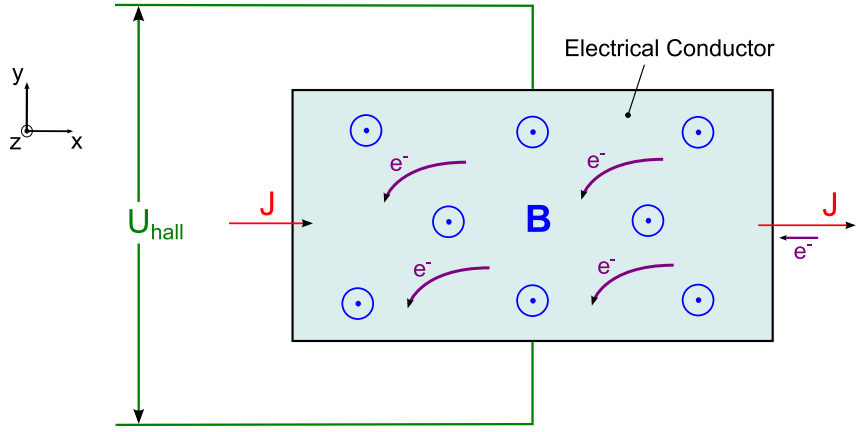

Figure 3.1: Hall Effect principle sketch. An electrical conductor is placed inside a magnetic field as indicated by the blue circled dots. A current J is driven through the conductor, resulting in a flow of electrons in negative x-direction. The velocity vector $\mathbf{v}$ of the electrons causes an acceleration on the particles given by the cross product $\mathbf{F} = q\mathbf{v}X\mathbf{B}$. As the charge q is negative, the force will be excerted downwards. As electrons gather in the bottom a net negative charge is built up, while the top gets an electron deficit which gives rise to a net positive charge. The resulting voltage U_{hall} is the Hall Voltage.

The Hall effect is the production of a voltage difference (the Hall voltage) across an electrical conductor as illustrated in figure 3.1. We have a magnetic field $\mathbf{B}$ perpendicular to the "flat side" of a conductor (positive z-direction). When we drive

a current through the conductor in the x-direction, there will be a flow of electrons in the negative x-direction. Because of the magnetic field, the electrons will be accelerated in the negative y-direction, giving rise to an electrical field in the positive y-direction. This results in the voltage difference $\mathbf{U}_{hall}$, the Hall Voltage. Note: For positive charge carriers instead of electrons, the effect will give the opposite Hall Voltage, given the same direction of positive current [30]. Advantages: As shown on figure 3.5, the Hall Effect technology can sense strong magnetic fields. A properly packaged Hall Effect Magnetometer is robust, and for this reason it is often used in applications like wheel rotation speed sensors. Disadvantages: Not very sensitive. Output is low voltage, and needs high amplification.

3.2 Flux gate

Flux gate magnetometers were invented in the 1930s by Victor Vacquier at Gulf Research Laboratories, and was used during World War II for detecting submarines. The flux gate magnetometer is based on what is referred to as the *magnetic saturation circuit*. As shown in figure 3.2 two bars of a ferromagnetic material are placed close together and wound with a primary coil in opposite directions.

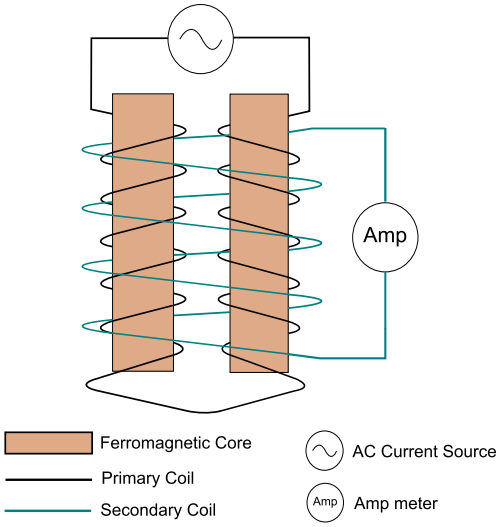

Figure 3.2: Principle sketch of a Flux gate magnetometer (inspired by [32]).

An AC current through the coils generates a magnetic field that will cause the cores to be magnetically saturated once every half-cycle. An external magnetic field will have one component in parallel with the two coils so that one of the cores will have its magnetization reinforced and the other coil will have it weakened by the same amount. This causes a time difference in magnetization between the cores, resulting in a net magnetic field. By winding a secondary coil around the cores, the field will induce a measurable voltage in this coil. Advantages: Good for precisely measuring DC fields. Has a relatively wide dynamic range. Disadvantages: Consumes a lot of power. Coils make these sensors physically large.

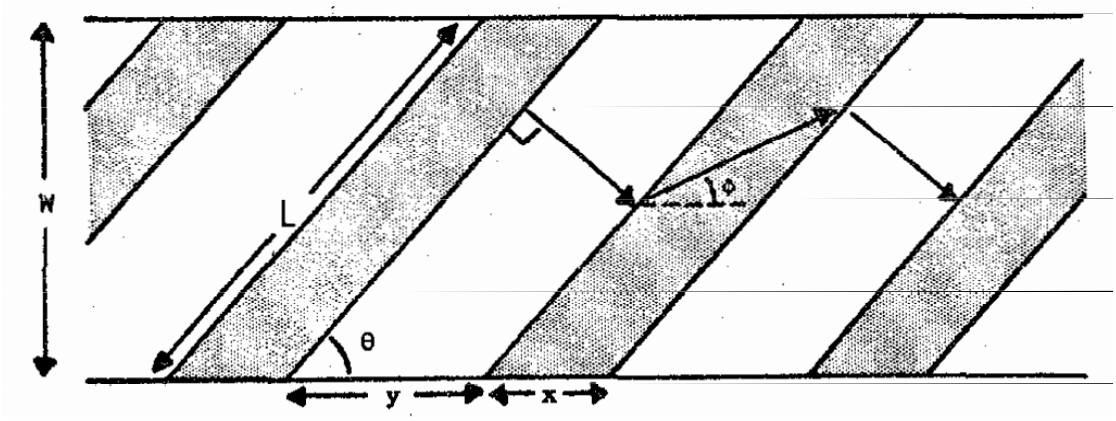

Figure 3.3: Barber Pole Structure(from [6]). The greyed out areas are high conductivity material, while the white areas are the magneto-resistive (MR) material. The structure makes sure that the current flows diogonally through the MR-material, at a 45° angle, relative to the magnetically aligned dipoles.

3.3 Anisotropic Magneto-Resistivity

Anisotropic Magneto-Resistive (AMR) material has the property that its resistance is dependent on the angle δ between the current flow $\mathbf{I}$ through the material and the magnetization $\mathbf{M}$ [7]. In polycrystals the resistivity is generally given as:

$$\rho = \rho_{\perp} + (\rho_{\parallel} - \rho_{\perp}) \cos^2 \delta \tag{3.1}$$

where $\rho_{\perp}$ is the resistivity in the material when $\mathbf{I} \perp \mathbf{M}$ and $\rho_{\parallel}$ is the resistivity when $\mathbf{I} \parallel \mathbf{M}$ [7]. Thus, the resistivity in the sensor element changes as a function of $\cos^2 \delta$. This function is linear when δ is close to 45°. For this reason MR sensors use something called a Barber Pole Structure [6].

The principle is illustrated in figure 3.3, where w is the height of the sensor element, perpendicular to the length. Essentially it involves introducing high conductivity bars between sections of the MR material at an angle of 45°, to alter the direction of the current flow through the material. It will now flow 45° relative to the length axis. When we discuss Set and Reset pulses in section 4.5, we will see that these pulses magnetize the dipoles so that they are aligned along the length of the element, so effectively the Barber Pole Structure ensures that the magnetization is biased 45° relative to the current flow. This Barber Pole Bias ensures that the sensor stays in the Linear Operating Region [23], as long as the applied magnetic fields are within the dynamic range of the sensor.

An AMR sensor element, as well as its functionality, is illustrated in figure 3.4. An application of such a sensor element is discussed in the description of the Honeywell HCM1043 3-axis magnetometer. Manufacturers of AMR sensors include Honeywell, NXP Semiconductors and Sensitec. Advantages: Large dynamic range. Miniatyrized, low-power components available off-the-shelf. More sensitive than Hall Effect sensors. Fast. Disadvantages: Low raw output voltage. Non-linearity property.

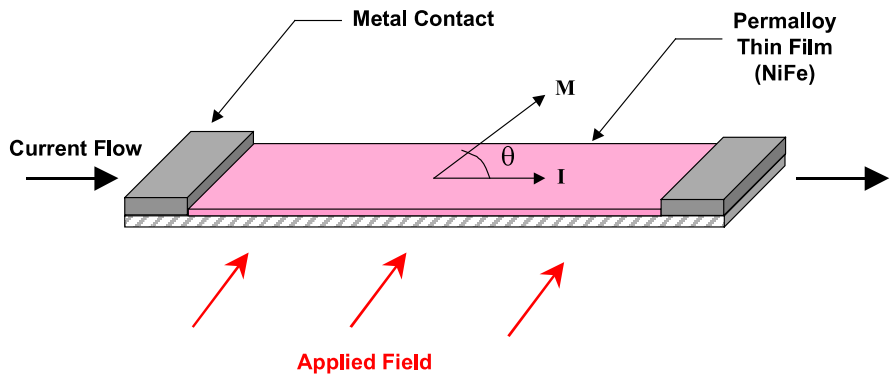

Figure 3.4: Simplified sketch of an AMR Element (from [21]). An applied magnetic field influences the angle θ between the magnetization M and the current I slightly, the angle still staying close to the 45° bias. The resistivity is then linearly proportional to the magnetization, and thus the applied field, as long as the applied field is within the linear range of the sensor.

3.4 Assessing sensor requirements

One of the goals of this project is to test a magnetometer for the CubeSTAR satellite. For this purpose, a sensor that is small, low cost, and can run on 3V or less would be ideal.

The earth's magnetic field has an intensity in the order of $300-600mgauss$, which means that the full range of the magnetic field to be measured will be in the order of $600-1200mgauss$ counting both polarities. Thus, if the sensor can measure with a resolution of about 100-200 $\mu gauss$, we can hope for a signal to noise ratio of around 6000:1, or $76dB$.

The given requirements for the SRADS magnetometer is to be able to measure signals with a frequency of up to 100Hz. The goal of the system will therefore be to measure all signals below 100Hz and filter out everything above that frequency. An analog lowpass filter with a passband of up to 100Hz needs an even higher sampling rate (oversampling) to be able to digitally filter out all potential aliasing (see section 7.9). The ICI-3 sounding rocket uses a Minor Frame rate of about 2.9kHz (see section 5.1) which will make a good sample rate for this application.

Based on the assessment of different magnetic sensor technologies (including the ones detailed in the previous sections) and the hardware requirements for the system, an AMR sensor seems to fit the purpose. The advantages of the AMR sensor in this respect are:

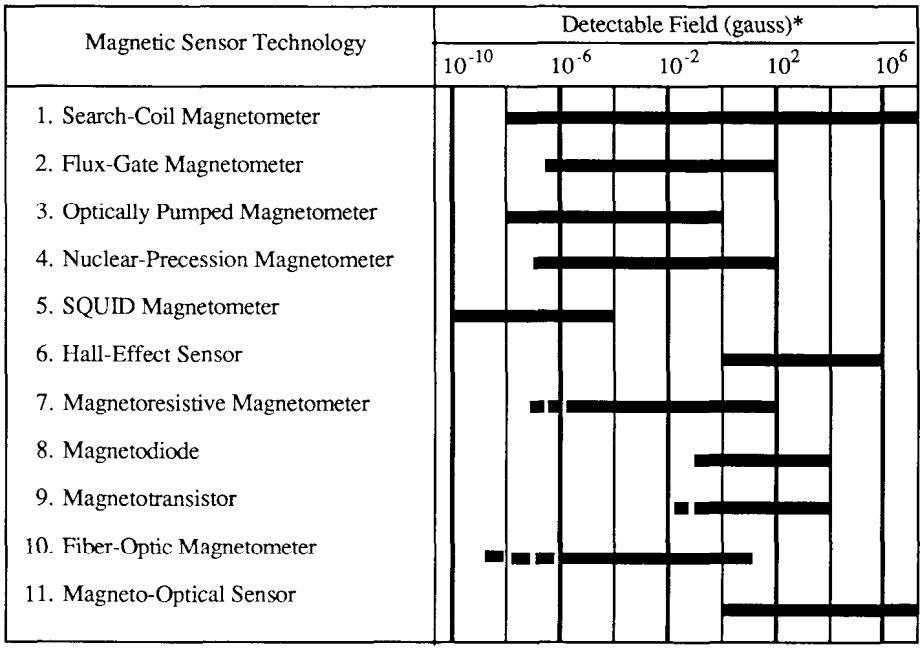

Figure 3.5: A comparison of different magnetic sensor technologies(from [9])

- Dynamic range is near-perfect for measuring earth's magnetic field, ranging from the order of $10^{-}6$ to 10^2 (see figure 3.5)
- Wide bandwidth, DC to 1Ghz [9]
- Solid state device entails small size, no moving parts, reliability and low power.
- High availability off-the-shelf parts

The Honeywell HMC1043 3-axis magnetometer is a good choice for a small, cheap magnetometer that meets the stated requirements. The stated bandwidth is 5Mhz, the resolution is given as 120 uGauss @50Hz and 5V bridge voltage.

Chapter 4

The HMC1043 Magnetometer

4.1 Principle of Operation

The Honeywell HMC1043 magnetometer is an Anisotropic Magneto-Resistive (AMR) sensor. AMR technology was introduced in section 3.3 on page 31. For its magnetic sensors, Honeywell uses sensor elements made from a ferrous material called Permalloy [25]. Four such elements are placed in a diamond shape with the ends connected by metalization to form a wheatstone bridge.

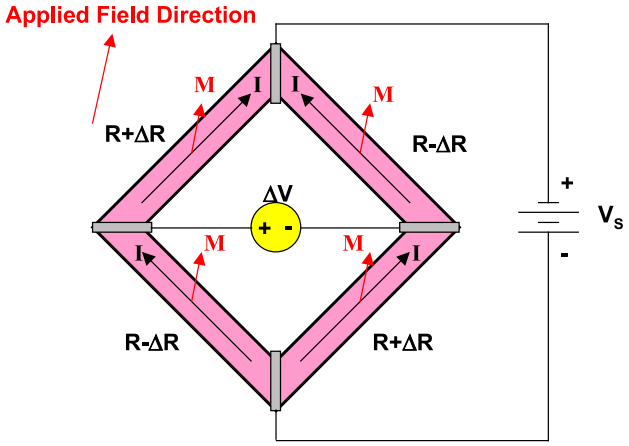

Figure 4.1: AMR Wheatstone Bridge (from [21]). Four AMR sensor elements are connected in a wheatstone bridge structure. A bridge Voltage V_S is applied across the bridge. As the resistance ΔR changes, so does the voltage output ΔV. As ΔR increases, the voltage $\Delta V+$ decreases and $\Delta V-$ increases, due to the continous (nominally equal) current through all the resistive elements.

As we can see in Figure 4.1, the top and bottom metals are connected to a voltage source, and the current continuously streaming through the bridge (from top to bottom) results in a voltage difference between the two side metalizations. This resulting voltage is proportional to the magnetic field present perpendicular to the bridge, and is available to measure on the respective $\Delta V+$ and $\Delta V-$ analog outputs of the sensor. The HMC1043, being a 3-axis magnetic sensor, has three such bridges oriented

perpendicular to each other, and so has three differential outputs, one for each axis. The sensor bridges are connected as shown in figure 4.2.

VB and VSS are the common voltage sources for all three bridges, and a voltage difference applied between these connections will cause the flow of current through the sensor elements. The OUT- and OUT+ connections for each bridge correspond to the side metalizations on figure 4.1, and are individually accessible from the pins on the outside of the chip.

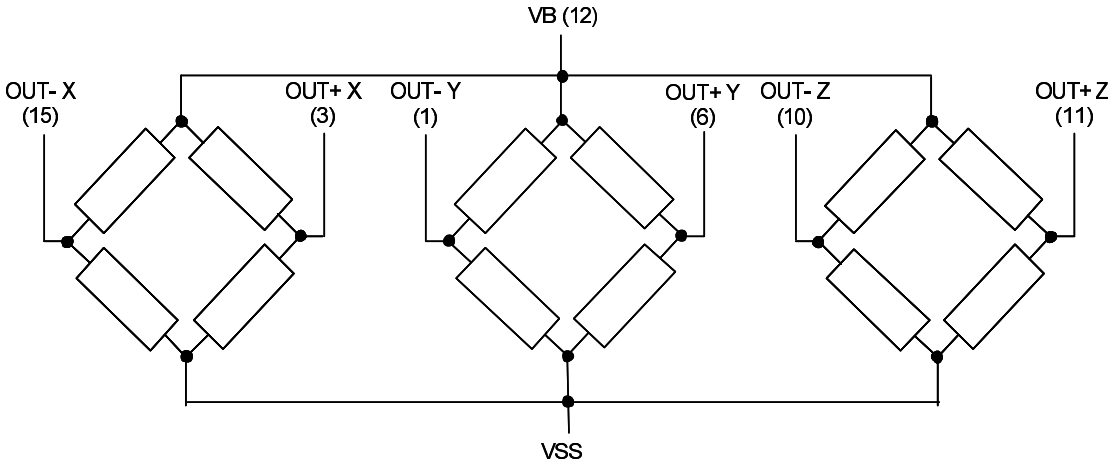

Figure 4.2: Schematical representation of a 3-axis Wheatstone Bridge configuration (from [20]). All three bridges share the same bridge voltage, but have individual outputs.

4.2 Physical Size and Pinout

The HMC1043 package is a 16-pin Leadless Plastic Chip Carrier (LPCC) measuring $3mm * 3mm$ with a height of $1.4mm$. On figure 4.3 we see the chip from the bottom side. We take note of the magnetic sensing directions, the x-axis pointing left, y-axis pointing upwards, and the z-axis pointing inwards through the figure. Picturing the chip looking down on the top side of the chip, oriented with Pin 1 in the lower right corner, we see that the positive x-axis points to the right, the positive y-axis points upwards, and the z-axis points straight at us outwards along the height axis of the chip (see figure 4.4).

We take note of the bridge voltage pins VB and VSS and the six positive and negative output connections for the three axes OUTnn. Offset pins OFFnn will be discussed in section 4.6 and SR pins for the Set/Reset strap will be discussed in section 4.5. ('Set/Reset' will hereafter be referred to as 'SR')

4.3 Voltage Output and Sensitivity

The sensitivity of the sensor is given as (typically) $S = 1.0mV/V/gauss$. With a bridge voltage of $VB = 2.5V$ the resolution of the magnetometer will be given by

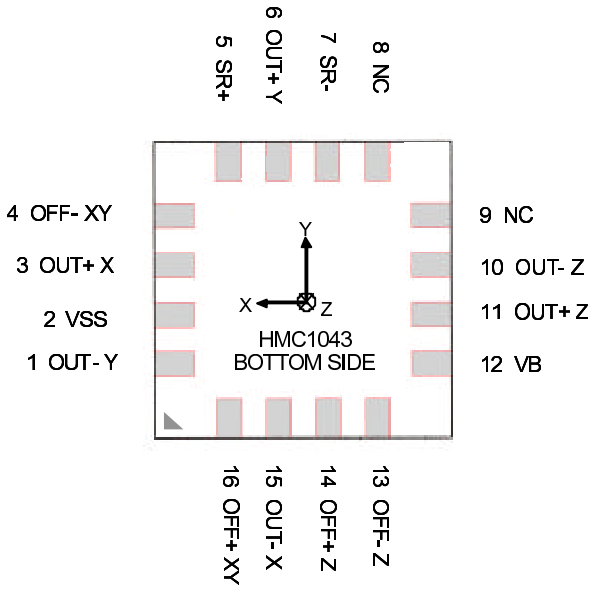

Figure 4.3: Pinout of the HMC1043 as seen from the bottom side (from [20]). The sensitive Z-axis points into the page, the X-axis points to the left and the Y-axis points upwards. We notice that Pin 1 is at the lower left.

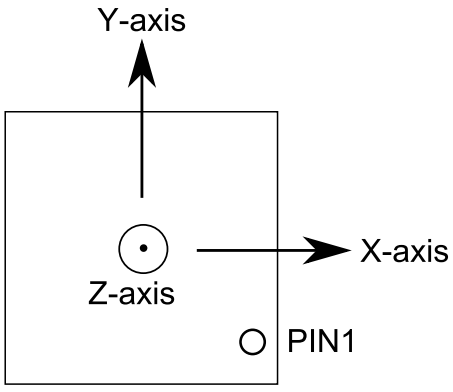

Figure 4.4: HMC1043 as seen from the top side. The sensor has been rotated 180° around the Y-axis with respect to the bottom view in figure 4.3. Pin 1 is now at the lower right.

equation 4.1.

$$S_{2V5} = S * VB = \frac{1mV/V}{gauss} * 2.5V = \underline{2.5mV/gauss} = \underline{2.5nV/\mu gauss} \tag{4.1}$$

The stated resolution of the magnetometer is $B_{delta} = 120\mu gauss$, so the smallest voltage step V_{delta} can be found using equation 4.2.

$$V_{delta} = S_{2V5} * B_{delta} = 2.5nV/\mu gauss * 120\mu gauss = \underline{300nV} \tag{4.2}$$

It is assumed that the sensor will be subjected to a maximum magnetic field strength of $B_{max} = 0.6gauss$. The expected full-scale voltage output can be calculated using equation 4.3.

$$V_{FS} = S_{2V5} * 2 * B_{max} * VB = 2.5nV/\mu gauss * 2 * 0.6gauss = \underline{3mV} \tag{4.3}$$

The Analog-to-Digital Converter will use a range of $VREF = 2.5V$ ($*2$ for a differential ADC), so the amplification A needed to take advantage of the full resolution of the ADC is given by equation 4.4.

$$A = \frac{2 * VREF}{V_{FS}} = \frac{2 * 2.5V}{3mV} \approx \underline{1667} \tag{4.4}$$

These calculations are based on the typical sensitivity, not maximum sensitivity, and as we are using a 16-bit ADC, we are not expecting to take advantage of the full resolution. Aiming for using around half that scale should yield satisfying results, so we will aim for an amplification of 800 from the sensor (Update: As the negative input of the ADC is locked at 1.25V, the dynamic range of the inputs is $\pm 1.25V$, therefore an amplification of 621 was chosen for the final design). If a single-input ADC is used opposed to a differential input ADC, we should aim for an amplification of 400.

4.4 Noise

The stated value of the noise density of this device is $50nV/\sqrt{Hz}$ at 1kHz, which entails a noise amplitude of $N = \sqrt{1000} * 50nV = 1600nV$ for a $1kHz$ signal and VB = 5V. It can be assumed that the absolute noise amplitude will be lower with half the bridge voltage, and the expected signal frequency is less than 100 Hz, which makes N a worst-case figure. Considering the voltage step resolution from equation 4.2, the worst-case scenario is that the noise will degrade the resolution by a factor of $1600nV/300nV \approx 5$, which is a pessimistic estimate. The low-pass filter implemented (see section 7.9) might also have a positive effect on this noise.

4.5 Set and Reset

As mentioned at the start of the chapter Honeywell AMR sensors are fabricated with Permalloy thin film. This ferromagnetic material has magnetic domains that can be oriented in a certain direction. When the material is exposed to strong magnetic fields these magnetic domains can orient themselves in arbitrary directions [23].

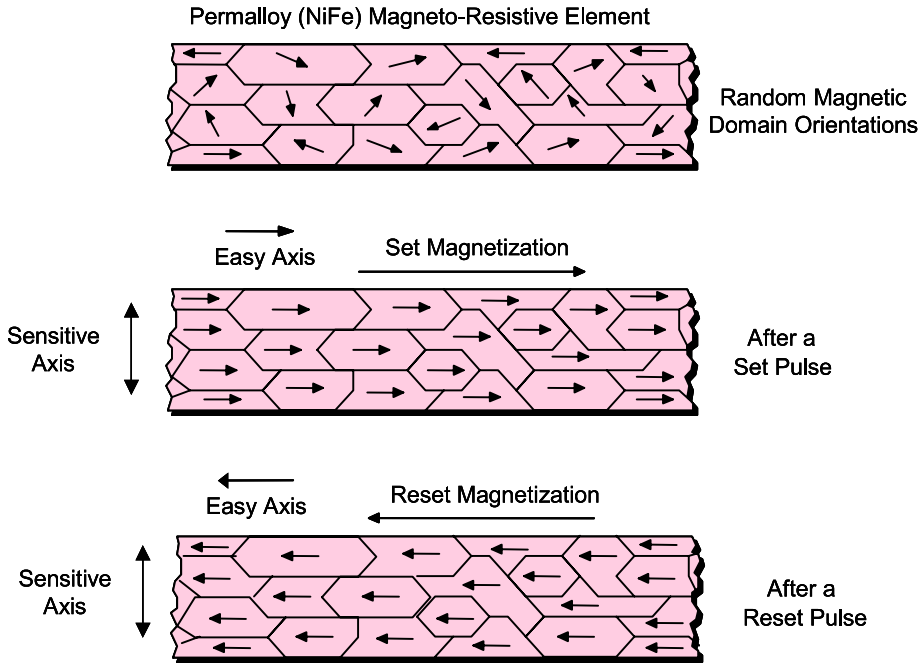

Figure 4.5: Magnetic Domains (from [23]) of the AMR sensor element. Before a Set or Reset pulse the magnetic dipoles of the ferromagnetic material are randomly oriented. A Set Pulse magnetizes the dipoles in one direction, while a Reset Pulse magnetizes them in the opposite direction.

This effect is illustrated in figure 4.5. The top illustration shows the randomly oriented magnetic domains after the magnetic domains have been disturbed. What we want is the magnetic domains to be unidirectionally oriented along the easy axis[1] of the ferromagnetic material, and perpendicular to the sensitive axis.

Honeywell AMR sensors have something called a SR-strap (See figure 4.6). This is a small coil that can be used to apply a strong magnetic field parallel to the easy axis of the AMR sensor elements, and will in that way orient the magnetic domains how we want them, namely parallel to the easy axis, 45 degrees relative to the sensing current and perpendicular to the sensing axis. This will ensure that the sensing element is at its most sensitive, and thanks to the barber pole bias discussed in section 3.3 the measured output voltage from the bridge will also be linearly proportional to the sensed magnetic field.

On the HMC1043 there are two such straps, and although it is not explicitly stated in the datasheet, we can assume that one is for resetting the x and y-axis elements,

[1] In materials science, the 'easy axis' refers to the energetically favorable direction of the spontaneous magnetization in a ferromagnetic material.

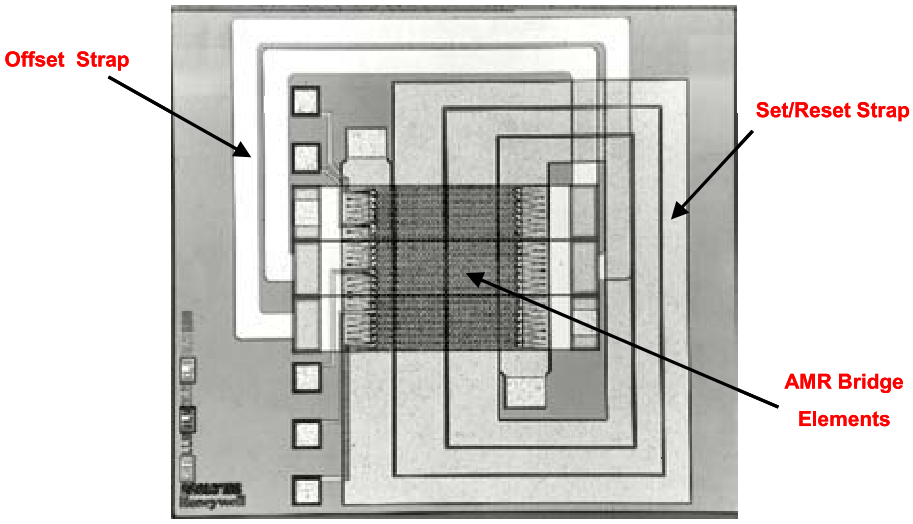

Figure 4.6: AMR die (from [23]). The reset strap runs perpendicular to the AMR sensor elements. When a current pulse is driven through the strap, the current (running vertically on the picture) will create a strong magnetic field (oriented horizontally on the picture), aligned with the length of the sensor elements, magnetizing the elements along their length.

while the second is for resetting the z-axis. This assumption is based on the general idea that the x and y-axis are more integrated as compared to the z-axis, and as we will see in section 4.6, this is also how the offset straps are organized. However, the positive and negative nodes for the two internal SR straps are connected together, so from the outside we see only SR+ and SR-, and for all intents and purposes it is treated as one strap. The resistance of the strap is typically 2.5Ω [20], and for a complete Set or Reset of the magnetic domains, Honeywell recommends driving a current pulse with peak value of $1A$ through the strap. This number has been changed to a typical $3A$ in the newest rev F of the HMC1043 datasheet, but [24] also states that $0.5A$ is sufficient for the HMC104x series, and the datasheet itself states $0.5A$ per strap under "Sensitivity". The design and results presented in this paper, indicate that a $1A$ peak current is sufficient. The minimum duration for the current pulse to stay at its peak value is $10 - 50ns$ for the range of different Honeywell AMR sensors. In section 6.2 on page 59, we will design the SR-circuit so that these requirements are met.

As the Set and Reset functions orient the magnetic fields opposite to each other, the voltage difference measured at the outputs will also have opposite signs while in Set and Reset mode respectively (See middle and bottom illustrations of figure 4.5).

4.6 Offset voltage

Another property of AMR sensors is the offset voltage difference present on the output when the measured magnetic field is zero. Figure 4.7 shows the offset voltage of around $2.5mV$ for an HMC1001 magnetometer. It is interesting to note that the graph for

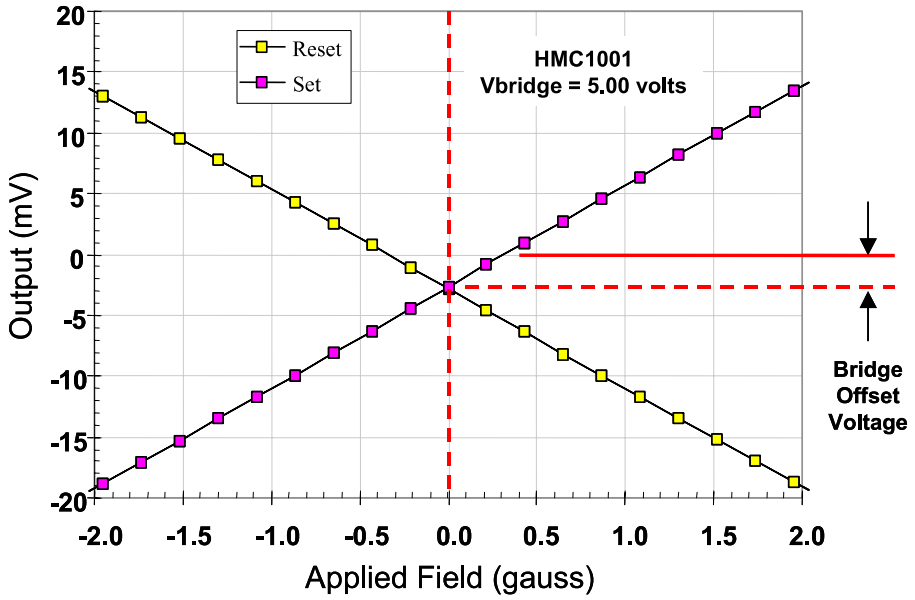

Figure 4.7: Output graph showing the wheatstone bridge offset of a Honeywell HMC1001 magnetometer (from [22]). We note that the graph for the sensor output after a Set pulse is opposite to the sensor output after a Reset pulse. However, it is not mirrored around $Y = 0$, but rather an offset from zero. When no magnetic field is applied to the sensitive axis of the sensor, the output voltage is the same for Set-mode and Reset-mode, and that voltage is the Offset Voltage.

the sensor in Set mode crosses the graph for the same sensor in Reset mode at the zero gauss line. This will prove to be useful when we discuss some of the methods of dealing with bridge offset.

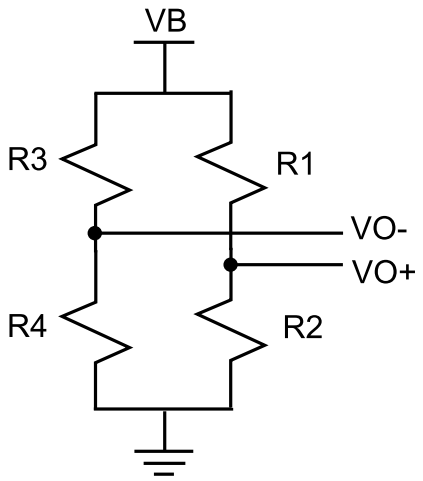

Figure 4.8: A principle sketch of one wheatstone bridge structure. The VB and ground nodes are the nodes to which the V_S voltage from figure 4.1 is connected, while the ΔV voltage measured between the side metalizations on that figure corresponds to the voltage measured across $VO+$ and $VO-$ here.

The offset voltage is a result of a mismatch between the resistance in the four

elements of the bridge. Taking a look at figure 4.8 it is apparent that the positive and negative outputs are given as a result of their respective resistors. When the sensor senses no external magnetic field, we can find the positive and negative Offset Voltages by solving equation 4.5 and 4.6 (ref [22])

$$VO+ = VB * \frac{R2}{R1 + R2} \tag{4.5}$$

$$VO- = VB * \frac{R4}{R3 + R4} \tag{4.6}$$

The offset for the HMC1043 is stated to be a maximum of $\pm 1.25mV$. The expected range of the sensor voltage output as we calculated in section 4.3 on page 36 is $3mV$, or rather $\pm 1.5mV$, so the offset voltage can actually be a significant component in the total voltage output. Considering the planned amplification of 400 (Update: 621 was used), the offset voltage would manifest itself as a worst case of $\pm 0.5V$ on the input of the ADC.

There are several ways to deal with offset voltage [22]. These will be presented briefly in the following subsections.

4.6.1 Offset Strap Current

Though not the easiest, this is the most straightforward and intuitive way to solve the offset voltage issue. It consists of driving a current through the offset straps found on the magnetometer, the OFFnn pins in figure 4.3. By applying the current, a magnetic field is created that sums with the external measured field to cancel the offset voltage. On the HMC1043 there is one offset strap for the z-axis element, and a combined offset strap for the x- and y-axis elements. The reason for this is assumed to be that the mismatch in the resistance values for the x- and y- axes are mostly identical. This may have something to do with the sputtering of the thin film Permalloy, which could be a process common for the x- and y-axis, while the z-axis is made separately, but this is only speculation.

An advantage to this method is that it is relatively easy to perform this function with little hardware, either using a potentiometer for manual trimming, or some kind of digitally controlled potentiometer or DAC for driving current through the strap. Another advantage is that it can also be used for annulling a strong continuous field nearby the sensor, or at least it could if all the offset axes had individual offset straps. The disadvantage of this method is that it either requires a manual calibration of each chip or some kind of logic procedure similar to the Digital Subtraction method in 4.6.2. It is unclear if the lack of individual offset control for the x- and y-axes is of any significance. As the offset straps will not be used, this issue will not be discussed further.

4.6.2 Digital Subtraction

The Digital Subtraction Method takes advantage of the phenomenon we noted on the graph in figure 4.7, namely that the output value graphs of the Set Mode and

Reset Mode cross each other in the zero gauss line. More precisely, for any measured magnetic field, a sensor in Set Mode will give the opposite voltage output as in Reset mode. That is; with an ideal zero offset sensor. With a real chip with offset, the "opposite voltages" will not be mirrored around the 0V line, but rather around the Offset Voltage line. Subtracting one output from the other will in fact produce the Offset Voltage, so that is exactly how the Digital Subtraction method is implemented. By continuously applying Set and Reset pulses to the SR strap and taking measurements in between, each measurement can be Offset-corrected by taking two measurements, one in Set Mode and one in Reset mode, and mathematically adjusting them to be centered around 0. It does not even have to be done often, if it is done once, the offset voltage can be calculated and be taken into account for all subsequent measurements.

The advantage to this method is that it requires no extra hardware, and is straightforward to implement. The disadvantage is what was pointed out in section 4.6. The offset voltage can be of considerable size, and constitute up to nearly half of the Full-Scale measurement region, up to $\pm 0.5V$ after amplification. This type of SR Switching will also make analog low-pass filtering difficult, as we will use a cutoff frequency much lower than the switching frequency (see section 7.9).

4.6.3 Shunt Resistance

This method builds on the theory presented with equation 4.5 and 4.6. It works by simply adding a resistor in parallel to one or more of the resistors in the Wheatstone bridge (see figure 4.8). The first step is to identify the larger resistor value and shunting that path with an external resistor.

The disadvantage to this method is that for obtaining accurate results the process of selecting and procuring correct resistor values is difficult and the calibration procedure must take place in an environment without any magnetic stimulus, such as within a Helmholtz cage, and the procedure must be repeated for each sensor. It will also not scale with changing offset as a result of e.g. temperature change. The advantage to this method is that it requires relatively little hardware to realize once the perfect setup for the sample chip has been found. In a production line with Automated Test Equipment, this method can be used effectively.

4.6.4 Amplifier Bias Nulling

As figure 4.9 shows, this method involves connecting an opposite polarity voltage to null out the offset voltage at the input of the amplifier stage. A potentiometer is used to control this voltage and must be trimmed to cancel the offset voltage of the sensor.

The advantage of this method is that it is relatively easy to implement. The disadvantage is that it will also need a Helmholtz cage or switching SR outputs in order to be calibrated, and calibration must be performed for each individual chip. It will also not scale with changes in offset. A workaround to this could be to analyse the measurements automatically, as with the digital subtraction method, and connect some digital logic to a programmable potentiometer. This could be a

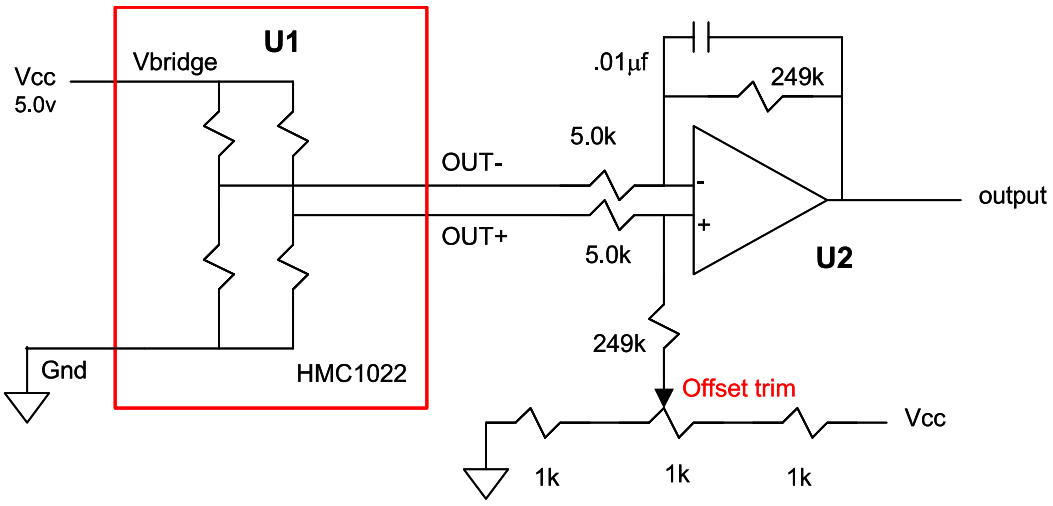

Figure 4.9: Amplifier Bias Nulling method (from [22]). A potentiometer controls the bias nulling voltage that is applied to the first amplifier input stage. The potentiometer must be trimmed so that the output from the first amplifier stage is rid of the bridge offset voltage.

numerical alternative to the Switching Feedback solution from section 4.6.5, but was not investigated at an early stage in the development.

4.6.5 Switching feedback

This method is probably the most versatile, but also the one that requires the most hardware to realize. It is in a way related to the Bias nulling method, in that it does not influence the actual bridge currents, but rather attacks the problem at the amplifier input, by forcing a bias voltage on the input to null out the offset. It can be seen as an automatic self-calibrating variant of the Amplifier Bias Nulling method from section 4.6.4. It also shares much of the same principle theory as the Digital Subtraction method from section 4.6.2.

By applying continuous Set pulses and Reset pulses so that the sensor stays in Set mode half of the time and Reset mode the other half of the time, the duty cycle of the output square pulse will be 50%. As we can see on figure 4.10, and for the same reason stated in the section on Digital Subtraction (4.6.2), this square pulse will be symmetric around the Offset Voltage V_{OFF}. By this follows that we can produce the offset voltage by integrating this square pulse.

A principle circuit for the Switching feedback circuit is shown in figure 4.11. Amplifier U3 in this circuit works as an integrator, integrating the output from the first differential amplifier stage (opamp U2). This integrator is inverting, so that the resulting bias on the input of the first amplifier stage will null out the offset voltage, in a similar way to the Amplifier Bias Nulling method. Because Vref is the zero-value reference of the integrator, the integrator will continuously adjust its "sum" as a result of the divergence of the net output voltage Vout1 from Vref. In its stabilized state, the biasing voltage will therefore settle to just the right value so that output Vout1

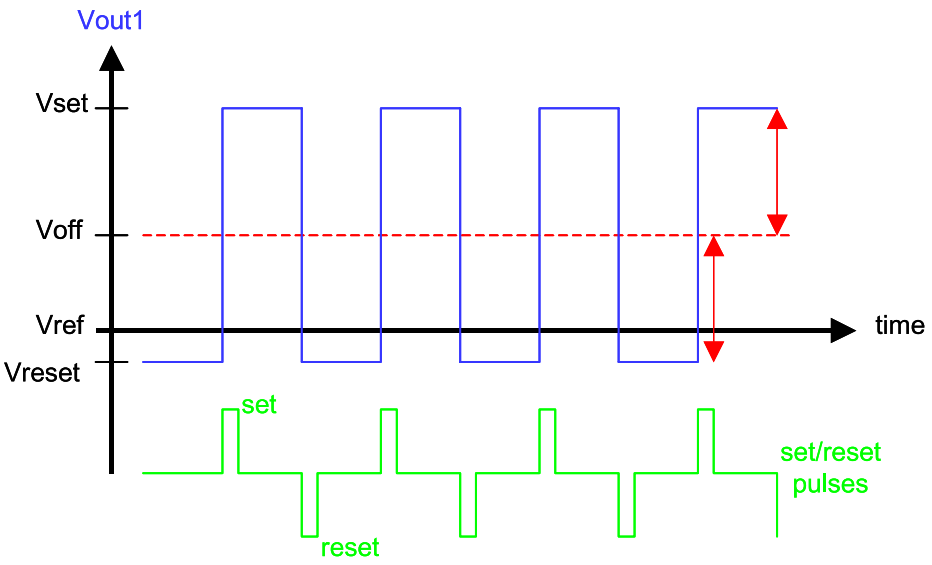

Figure 4.10: SR Switching Output (from [22]). As Set and Reset pulses are applied periodically with equal distance between them, a 50% duty cycle square pulse voltage can be seen at the output from the sensor. The average value, or the voltage level $Vout1$ around which they switch, is the Offset Voltage.

will stay swinging centered around the value of Vref, and the net value of Vout1 will be Vref.

The final stage of this circuit is the demodulation of the square pulse, which is done by synchronizing the amplifier U4 to the digital SR signal. Half of the time, when the non-inverting input is connected to Vref, the amplifier works as an inverting amplifier with unity gain. The other half of the time both the inverting and noninverting input of the amplifier are connected to the same signal, Vout1. This means that the voltage on the negative terminal will stay the same as Vout1 and so the (ideal) amplifier will not amplify or weaken the signal. In reality there is a small input bias voltage on the opamp that will cause a small change in the voltage on the output, but this is negligible because the signal has by then already been amplified. More on this in the sections on concept testing.

The disadvantage to this method is that it is the one that requires the most hardware, and consumes the most power. It may also be the most complex, but is by far the most interesting to investigate. The clearest advantage to this method of dealing with offset is that it is versatile and, once developed, will automatically calibrate to null any offset for any given chip. It does not require a magnetism-free environment for calibration, and will also automatically make adjustments to potential temperature-induced offset changes. I will make the case that there is a relatively high ceiling on how well this method can be implemented and, given enough time and study, it may hold the key to the most robust and reliable magnetometer implementations. We shall see my attempt at implementing this circuit in the hardware design section 6.3.

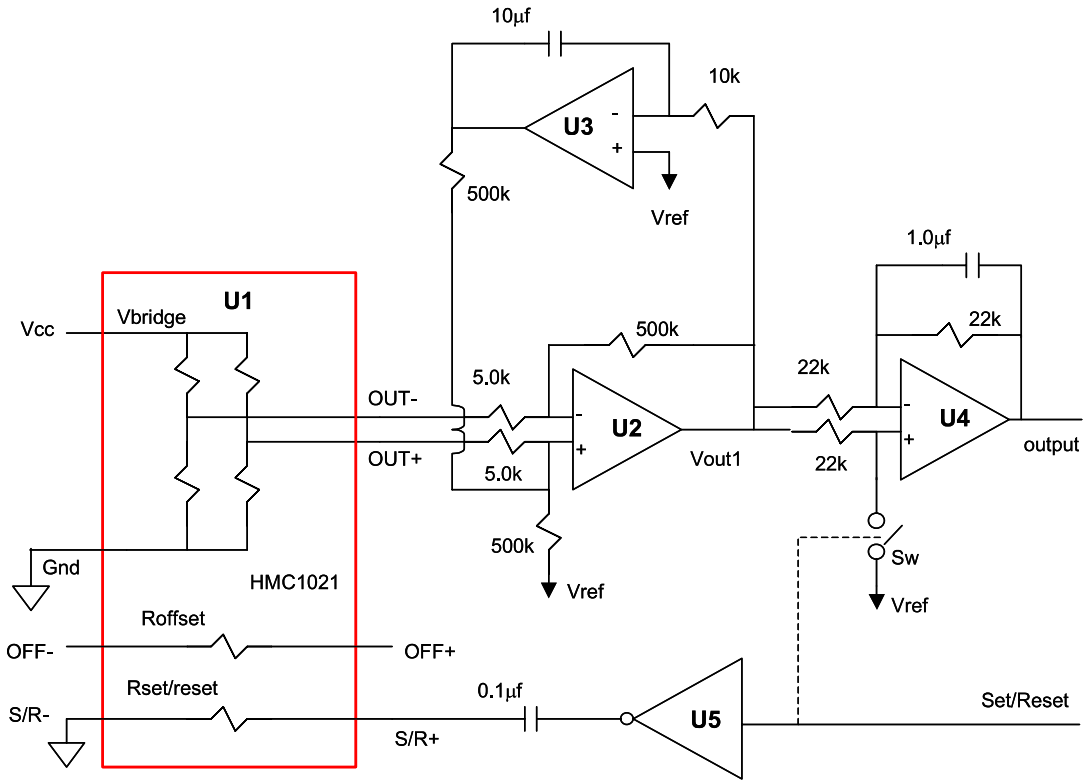

Figure 4.11: Switching Feedback Circuit Principle(from [22]). A square pulse seen on the output Vout1 is integrated at the amplifier U3, and fed back to the input of amplifier U2. After the feedback loop has settled, the resulting square pulse output from Vout1 will be centered around the reference voltage Vref. The signal is then demodulated in amplifier U4 to create a smooth output.

Chapter 5

Digital Control

This chapter will discuss the digital control section of the system. This subsystem will perform a multitude of functions.

- Communication with the ICI-3 onboard encoder
- Timing of the digital SR signal (see section 4.5)
- Timing of the demodulation-signal for the Switching Feedback Circuit (see section 4.6.5)
- Sample timing and communication with the ADC
- Auxiliary I/O for debugging
- Optional UART transmission

The following sections will discuss these points and what requirements they put on the digital system.

5.1 ICI3 PCM Encoder

5.1.1 Overview

The interfacing and communication with the ICI-3 onboard encoder is defined in the Hotel Payload interfacing document provided by Andøya Rocket Range [17]. This is a document dealing with the general aspects of the PCM encoder. The encoder interface supports both digital and analog modules, but only the digital module will be discussed here. Format sheets are also provided [19] which detail how the encoders in the ICI-3 rocket will be set up. The communication is performed as a synchronized serial communication, where each instrument has a defined timeslot where it can send its data. The timeslots are organized into major frames, minor frames, words and bits. 8 bits constitute a Word, 144 Words (1152 bits) constitute a Minor Frame, and 64 Minor Frames (73728 bits) constitute a Major Frame. By careful timing using the control signals from the encoder, the instrument drives the DATA signal to send its digital data to the encoder. The control signal outputs from the encoder are:

- SCLK: Serial Clock
- GATE
- MINF: Minor Frame
- MAJF: Major Frame

The Serial Clock is the heartbeat of the communication system. For ICI-3 the frequency of the serial clock is 3.333333 MHz. This means that we have a Word rate of 416.7 kHz, a Minor Frame rate of 2893.5 Hz, and a Major Frame rate of 45.2 Hz. Each complete cycle of SCLK amounts to one data bit in the communication format. All other control signals (GATE, MINF and MAJF) are updated by the encoder at the Falling Edge of this clock. This means that we can use the Rising Edge of SCLK to read the control signals (half a clock cycle later). The DATA signal will be read by the encoder just before the Rising Edge of SCLK.

5.1.2 Frame Format

The ICI-3 rocket has two onboard encoders, TX1-main and TX2-slave. The two SRADS MAG instruments will be connected to one encoder each, and as such have different timeslots to adhere to. The reserved time slots in the TX1-main and TX2-slave encoders are shown in tables 5.1 and 5.2 respectively. For both encoders, three timeslots of two words have been assigned. In other words, each Minor Frame the instrument can send three 16-bit values to the encoder. As we are going to use a 16-bit ADC to sample measurements from three axes simultaneously, this means that we can take samples once every Minor Frame and send all the data to the encoder during the same frame. This provides a sampling rate of 2893.5 SPS - the same as the Minor Frame rate.

5.1.3 Control signals

As mentioned in section 5.1.1 there are, in addition to SCLK, three control signals that the encoder uses to communicate with the instrument. We will here introduce briefly what their function is.

MINF and MAJF

The Minor Frame signal is driven high for one clock cycle at the start of every new Minor Frame. The exact details of this timing is not crucial for correct operation of sending the measurement values in this project, because the transmitter will incorporate a GATE-driven data driver to transmit the data (see *GATE*). The assertion of MINF happens at the beginning of the first word in each Minor Frame, and we can

Word									
0	Sync	Sync							7
8									15
16									23
24									31
32					MSB	LSB			39
40									47
48									55
56									63
64									71
72									79
80					MSB	LSB			87
88									95
96									103
104									111
112									119
120									127
128					MSB	LSB			135
136									143

Table 5.1: TX1-Main Minor Frame (simplified)

Word									
0	Sync	Sync							7
8									15
16									23
24									31
32									39
40					MSB	LSB			47
48									55
56									63
64					MSB	LSB			71
72									79
80									87
88					MSB	LSB			95
96									103
104									111
112									119
120									127
128									135
136									143

Table 5.2: TX2-Slave Minor Frame (simplified)

read out the signals using the Rising Edge of the SCLK-signal. The MAJF signal is used to signify the start of a new Major Frame and will be driven in the same way as the Minor Frame signal at the start of the first word in the Minor Frame, except not every Minor Frame, but only the first Minor Frame in each Major Frame (every 64 Minor Frames). Both the MINF and MAJF signals can be used to drive Minor Frame and Major Frame counters respectively, which can be used by the digital logic to keep track of time and be synchronized with the encoder. They can be used to initiate measurements that should occur at specific times, reset registers, or prepare the right data to be sent at the right timeslots.

GATE

This signal is used to control the readout of data from the instrument. While the Minor Frame and Major Frame signals are used for preparing the data to be sent, the GATE signal is the signal that actually initiates the transfer of data. Figure 5.1 shows the principle of the GATE signal operation.

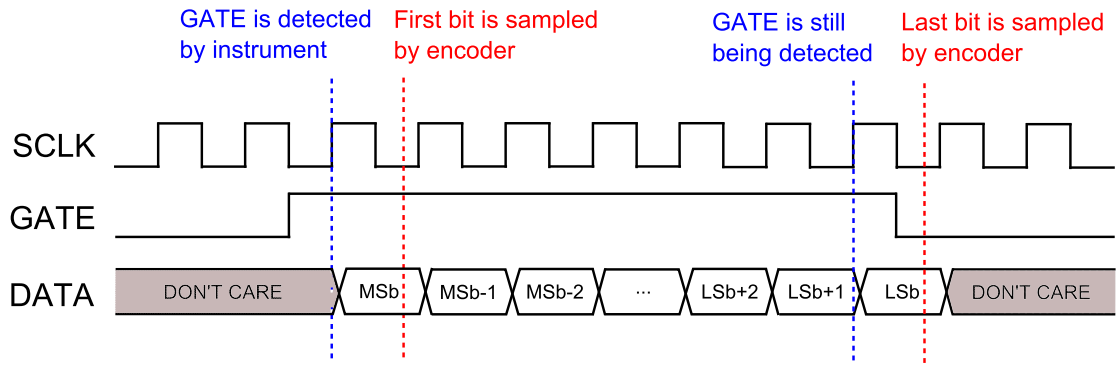

Figure 5.1: GATE timing [17]. As the GATE signal goes high, there is between one and 1.5 clock cycles until the first data bit is sampled at the encoder. When a high GATE signal is detected on the rising edge of SCLK, we can make the first data bit ready for sampling.

The instrument should check for a high GATE on the rising edge of SCLK and react to it by making the first bit available for transmission. The first bit will then be sampled by the encoder within one clock cycle, as illustrated on the timing diagram. The subsequent bits shall be shifted out as long as the GATE signal stays high.

5.2 Timing requirements

In the previous section we saw that the ICI3 encoder employs a custom interface, documented by specific formats and timings. Especially the GATE signal requires extensive control to handle correctly, and a programmable logic device and VHDL code are suitable tools to deal with this kind of task.

The SR control signal benefits from exact timing control, and knowing that we will be synchronized with the on-board encoder, it is only natural that we derive

these timings from the serial clock and communication formats. The demodulation signal should most likely be in synch with the SR signal (see section 4.6.5) but, as the sensor output voltage might be somewhat delayed in respect to the actual SR pulse, it is beneficial to have the ability to skew the demodulation signal by a number of clock periods in respect to the SR signal.

The time of the sampling/data conversion should also be synchronized with the encoder formats, and preferably done at a time when the electrical noise in the system, such as ground noise produced by switching during MINF flanks, is at a low. For that reason it will be crucial to be able to have accurate control over this timing.

All in all, timing requirements suggest that a programmable logic device such as a CPLD or FPGA will be suitable for the task.

5.3 Arithmetic/Procedural requirements

The demands on the arithmetical abilities of the digital logic will be low. The task of this system is to synchronize itself to the encoder, control sampling via binary signals, communicate with the ADC, and shift data bits out. Little to no mathematical operations will be done on the data in logic. These functions can be done by a microcontroller and a CPLD equally well.

5.4 Communication

Communication with an ADC is normally straightforward with a microcontroller. Hardware SPI/I_2C blocks make this type of task simpler to realize than in VHDL code. As we shall see in the sections 6.1 and 7.5, communication with the ADC is generally well documented and feasible in a VHDL-environment. The Auxiliary UART transmission would require almost no work to realize in a microcontroller, and clearly favours that choice in digital logic, though a custom "built from scratch" UART provide interesting opportunities like customizing bitrate.

5.5 Input and Output

For the realization of this design, we need to consider the minimum requirement for the number of I/O-signals. For the PCM-interface, five logical signals are needed. We will be communicating with three Analog to Digital Converters (ADCs), one for each magnetometer axis, and considering a total of three pins per ADC, we are up to nine signals. The Set/Reset signal needs one signal, the demodulation signal (see section 4.6.5) needs at most one signal for each axis, so *there* are an additional four required signals. The bare minimum number of digital signals needed for this design is therefore $5+9+4 = 18$. Reserving some signals for auxiliary I/O ports for alternate communication, debug-signals and/or jumpers/buttons/switches, a good estimate for signals needed might be 30. Most CPLDs and Microcontrollers available today offer this amount of I/O ports, so this does not favour one or the other technology.

5.6 Programmable logic

Taking the discussion of the previous sections into account, the choice falls on a programmable logic device. This choice is reinforced by successful earlier implementations of interfacing the same type of encoder using programmable logic devices [5] [2]. In these cases the Altera MAX and MAXII devices were used, and taking advantage of this past experience suggests using a similar device as a starting point.

5.7 Altera MAX II Development board

The Altera MAX II Development Board (hereafter referred to as the devboard) is a platform for evaluating MAX II features and prototyping CPLD designs. At the heart of the board sits the EPM1270F256C5N chip, which is a MAX II CPLD with 1270 logic elements in a 256-pin FBGA package. Among the features of this board are: A 66Mhz external oscillator that can be used for clocking the CPLD, a JTAG interface, 10 Multi-purpose LEDs, and many of the MAX II I/O pins are accessible through Altera's Expansion Prototype Headers[1]. All in all, this is a suitable board for exploring the MAX II architecture.

5.8 The Max II device family

All the MAX II family devices are compatible in terms of features, and although the initial development was done on the MAX II development board with a general MAX II chip, the principles of operation are the same. The family has smaller devices and both the MAX II G and the MAX II Z devices were considered, the main differences being that they draw less power and have a smaller footprint than the original.

5.8.1 MAX II Z

The MAX II Z was the first device to be considered. It is available with 240 or 570 Logic Elements (LEs), which both should be enough for this application. It comes in 68 or 100-pin Micro Fineline Ball Grid Array (MBGA) packages which are tiny packages with footprint areas of $5mm * 5mm$ and $6mm * 6mm$ respectively, while the pads are configured in a ballgrid with $0.5mm$ spacing. The main advantage of this chip compared to its bigger brothers, aside from the size, is that it consumes very little power. The idle current stated from the manufacturer is $25\mu A$. However, it was decided not to use this chip based on the fact that card production might prove unnecessarily challenging, and the space constraints on this project are not that tight (see section 7.1).

[1] One 40-pin, one 20-pin and one 14-pin standard double pinrow connectors

5.8.2 MAX II G

The chip chosen for the design would be the MAX II G. With a stated $2mA$ of supply current, it is not exactly power hungry, and the packages available for this chip are bigger and easier to handle when it comes to card-production. It comes in several different packages, but the 100-pin Thin Quad Flat Pack (TQFP) was chosen for its relative small size, measuring $11mm * 11mm$. This package is available with 240 or 570 LEs which should be more than enough, and with a maximum of 80 I/O-pins available it is more than versatile enough for the task. To make sure we have some headroom in terms of logic in the CPLD, a 570 LE chip would be good, and the actual chip used in the final design was the EPM570GT100C5N, which means it is a 570 LE MAX II G in a TQFP100 package, commercial edition with a speed grade of 5. Speed grade is not a real issue in this design, as all speed grades are more than quick enough to handle the clock speeds we will be operating at; a maximum of 5.5Mhz (see section 7.4).

Chapter 6

Concept testing

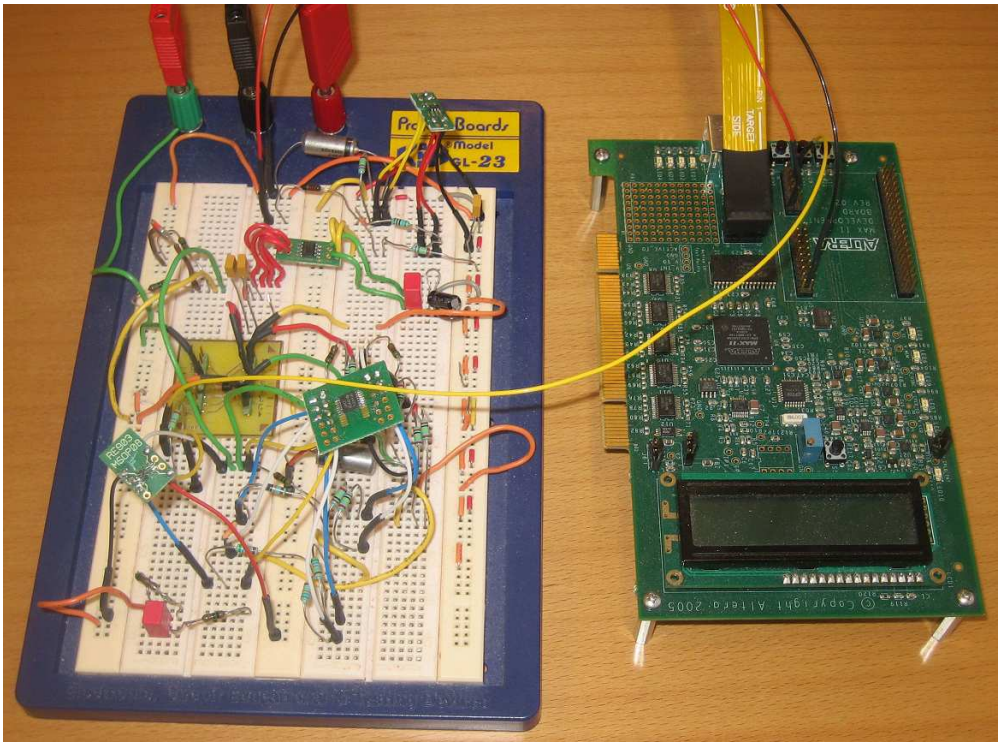

Figure 6.1: Test setup with Altera MAX II Development board and a breadboard. The breadboard contains an HMC1043 on a small breakout board, the SR circuit and the Switching Feedback Circuit.

This chapter was written to give an insight into the process of the early development and concept testing. The focus was on four important aspects of the system, the CPLD, interfacing of the ADC, the SR circuit and the Switching Feedback circuit. The CPLD naturally takes part in testing the other three, so it will not be tested separately.

6.1 ADC

The ADC used during the initial development was the LTC1864L from Linear Technology, a 16-bit 150kSPS single channel differential input ADC. Some features of this device are:

- Small package (MSOP8)
- Accepts low supply voltage (min 2.7V)
- Supports a serial clock of upto 8MHz
- Low power consumption, Typical: $450\mu A$, Shutdown: $10\mu A$
- Easy interfacing with SPI
- *True* differential inputs[1]

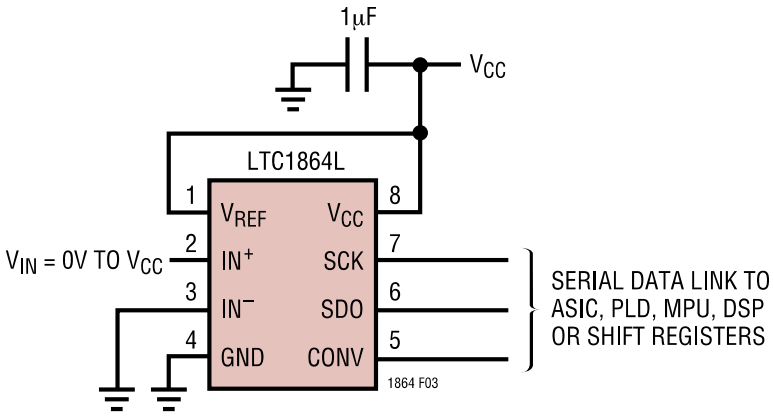

Figure 6.2: LTC1864 simple application, from [28]. As IN- is tied to GND, IN+ functions as a single-ended input. As V_{REF} is tied to V_{CC}, the dynamic range is 0 to V_{CC}.

The initial testing of this device was performed using a typical application circuit found in the datasheet of ADC (figure 6.2). Connecting V_{REF} to 3V gives the ADC a range of 3V on the inputs. As the IN- pin is tied to GND, the measuring range of IN+ is 0-3V. The IN+ pin was simply connected to a power supply with an adjustable voltage source. A 2.2uF tantalum capacitor was connected in parallell with the 1uF ceramic capacitor. There are two digital input signals, CONV and SCK, and one digital output signal SDO. They were connected directly to the CPLD via the expansion prototype headers. The functions of these pins are illustrated in figure 6.3. The CONV signal is the enable signal of the ADC. When it is pulled high, a conversion is

[1]This would turn out to be the disappointment of the device. Although the datasheet states *True differential inputs* under Features, upon closer examination, it does not actually support negative differential inputs (IN+ lower than IN-). The part was eventually replaced by a pin-compatible device from Analog Devices, AD7864 (more on this issue in section 7.5)

initiated. After a time of $t_{CONV} = 4.66\mu s$(maximum) the conversion is complete and the data is ready to be transmitted. Pulling CONV low at this point initiates the transmission and every falling edge of SCK will now shift out a new bit, MSB first. After the last bit has been shifted out, subsequent falling edges of SCK will produce 0's. Stopping the SCK switching will put the SDO into a high impedance state, and the cycle is complete. As we can see, it is possible to take advantage of a built-in sleep mode by delaying the transfer more than t_{CONV}. Taking full advantage of this behaviour can lower the current consumption slightly, but that has not been a focus in this work.

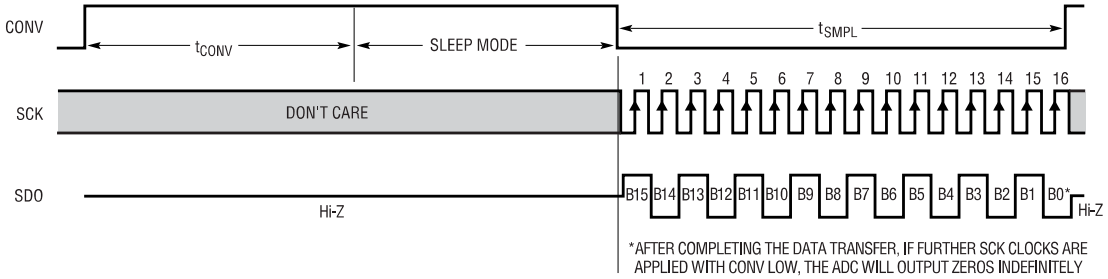

Figure 6.3: LTC1864L communication interface, from [28]. Applying a rising flank on the CONV signal initiates an analog to digital conversion. After a defined amount of time t_{CONV}, a falling flank on CONV together with applied serial clocks starts the data transfer.

A block diagram of the VHDL code used to test this setup is shown in figure 6.4. The complete code is displayed in appendix A. The *mclk* input is the 66Mhz master clock. This is divided by 20 to produce *clk*, the 3.3 MHz system clock. The *ADC state machine* is the main control block, and handles the startup-phase, producing the asynchronous reset signal *rst*, which is asserted at startup to reset all registers to known values. This state machine also controls the ADC-communication. A customizable sample timer signals the state machine to initiate a sample cycle by asserting *start_conv* for one clock cycle of clk. The state machine then goes through the stages of the communication with the ADC (see adc_ctrl.vhd in appendix A for details). The SDO signal is connected directly to a 16-bit shift register that is synchronized to the SCK clock controlling the ADC. In that way, the data is shifted in automatically. When the data has been completely shifted into the register, the state machine signals the rest of the system by driving the signal *new_data* high for one clock cycle. The shift register is transparent internally, so that all the data bits can be accessed simultaneously. In this test system, ADC functionality is tested by visualising the measurement on the multi-purpose LEDs of the devboard. The four most significant bits of the sampled data are connected to four of the LEDs.

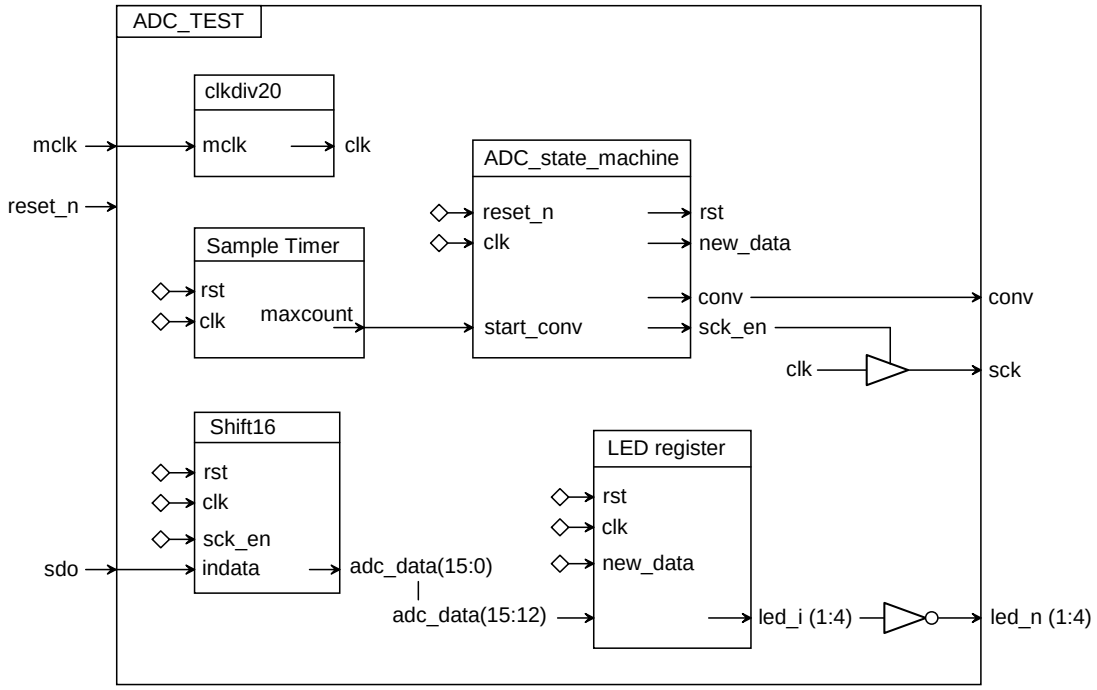

Figure 6.4: The block diagram shows the main building blocks of the ADC test code. The ADC State Machine block controls the ADC timing signals according to the timing diagram in figure 6.3. The SDO output from the ADC is shifted into a 16-bit shift register and the 4 most significant bits are displayed on four of the LEDs on the development board.

6.2 Set/Reset circuit

6.2.1 Required current/voltage

In section 4.5 the need to Set and Reset the magnetic domains of the HMC1043 sensor was discussed. A current with a peak value of $1A$ must be driven through the SR strap between SR+ and SR-. Considering we have 3V to work with, the first question to assess is: Will we get a current pulse with a high enough peak value using 3V? To answer this question, the maximum resistance of the SR strap must be considered. The SR strap of the HMC1043 has a nominal resistance of 2.5Ω, but the maximum value is 3Ω. The good news is that $0.8A$ is stated as the minimum pulse peak value. As the SR cycles will be run at high frequency of about 2.9kHz, it should give the sensor good and stable sensitivity.

6.2.2 Circuit principle

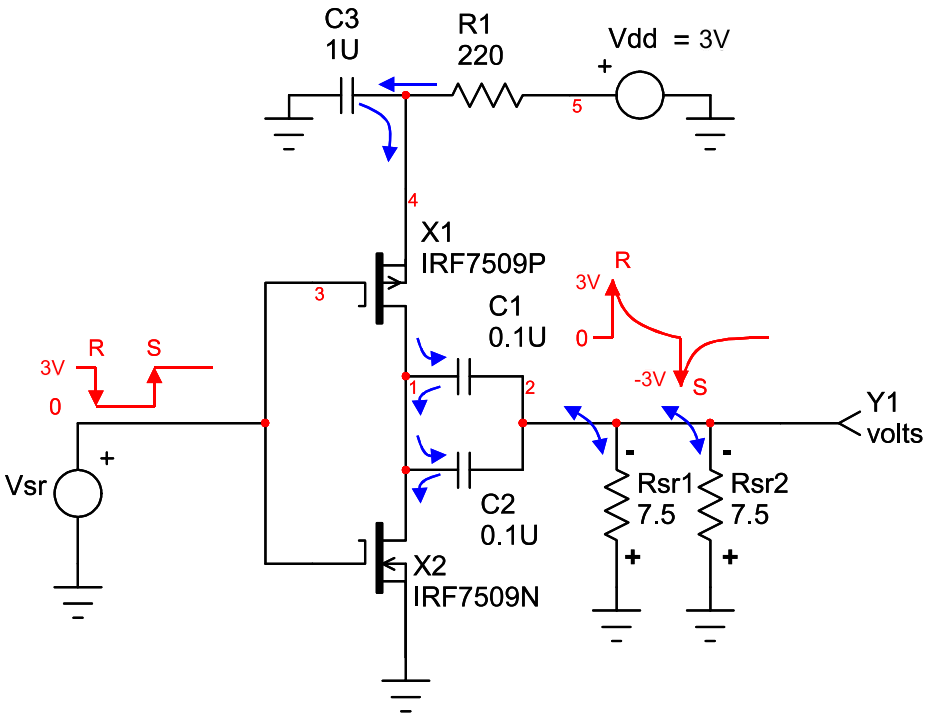

Figure 6.5: Totem Pole SR circuit principle based on figure [23]. The figure was modified to use 3V instead of 5V for supply and logic voltages, to reflect the tests performed during development. As the voltage in point 1 is switched synchronized with the gate voltage in point 3, current is pumped through the capacitors C1 and C2.

The Honeywell application note on SR functionality [23] gives a few suggestions on SR circuits. The "Totem Pole" circuit illustrated in figure 6.5 shows how both Set and Reset functionality can be achieved using MOSFET transistors connected as a cMOS inverter to pump charge through the SR strap (Rsr1 and Rsr2). A square

pulse is applied at the common gates of the nMOS (X2) and pMOS (X1) transistors. When the gate voltage is high, the nMOS is open and the pMOS is closed, resulting in a negative charge build-up (0V) in capacitors C1/C2[2]. The moment the gate voltage switches to low, X1 is quickly opened while X2 closes, effectively shorting the negatively charged C1/C2 and the positively charged capacitor C3. C1/C2 will quickly discharge through transistor X1 resulting in a sudden pull of negative charge into C1/C2 from the right side and, as a result, a current pulse will be driven through the SR-strap. Capacitors C1/C2 will then settle with positive charge and stay charged until the gate voltage changes once again. When the gate voltage switches back to high, X1 will close and X2 will open, resulting in a short between the positively charged C1/C2 and GND. Negative charge is pulled from GND through X2 and pushed through capacitors C1/C2 resulting in a current through the SR strap the opposite way. The capacitors once again settle with their negative charge, and one SR cycle is complete.

6.2.3 PSpice simulation

To verify the concept behind the SR circuit, a schematic as shown in figure 6.6 was drawn and simulated in PSpice. The Magnetometer SR strap is modelled as a resistor with a worst-case value of 3Ω, which is the stated maximum. As we will see in section 6.2.5, to ensure quick switching an extra cMOS inverter stage is employed. Spice models were downloaded from the International Rectifier website, to apply a degree of realism to the simulations. The goal however, was to test the circuit principle and get insight into values which are complicated to measure in real-life component tests, like the switching current through transistor M15. The stimulation file VSR.stl controls the programmable voltage source, and its content is shown in 6.1. It starts out and stays at 0V until $1\mu s$ has passed. It then rises to 3V with a rise time of $1ns$ and stays there until the $7\mu s$ mark. It then falls to 0V again with a fall time of $1ns$. In this model the C3 "charge battery" capacitor from figure 6.5 serves no function and is omitted, as the Voltage source in PSpice is ideal and can supply current infinitely fast.

```
1 +    (              0, 0)
  +    (     1.000e-6, 0)
  +    (     1.001e-6, 3)
  +    (     7.000e-6, 3)
5 +    (     7.001e-6, 0)
```

Listing 6.1: Stimulation file VSR.stl

Looking at the simulation result in figure 6.7, we see that the first current pulse reaches a peak value of $2.56V/3\Omega = 0.85A$. The expected time constant for the current pulse, the time when it reaches a factor 1/e of its peak value, is given by equation 6.1.

[2]As capacitors C1 and C2 are placed in parallell, they can essentially be considered as one capacitor of twice the size.

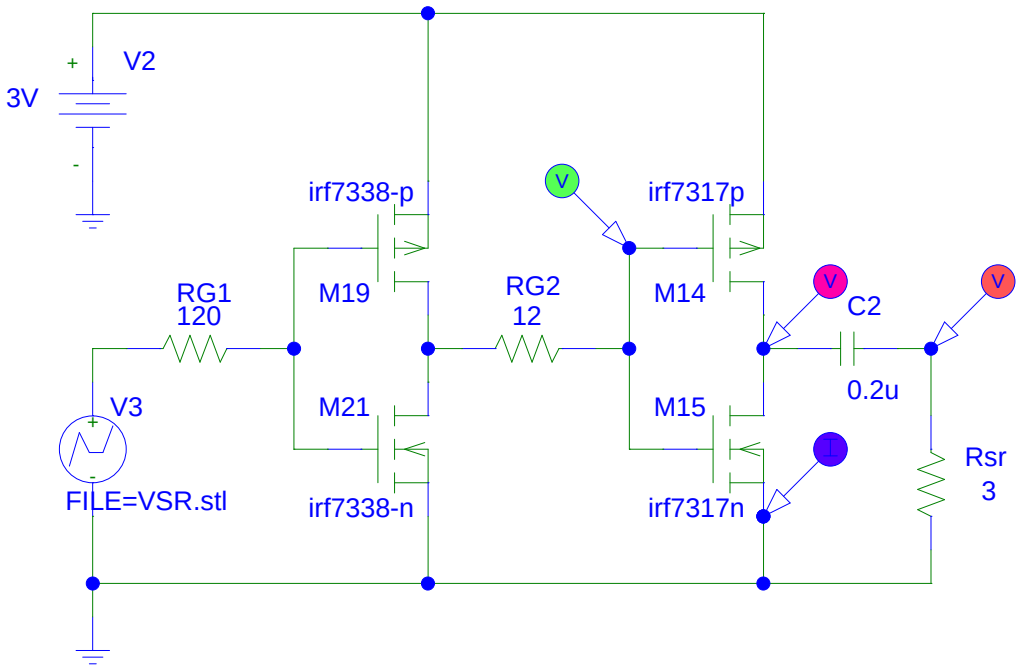

Figure 6.6: The circuit used to test the totem pole circuit principle in PSpice. A double inverter stage is employed to make sure that the gates of the power-transistors M14 and M15 can be charged fast enough for a sharp current pulse. The SR strap is modelled as a resistor $Rsr = 3\Omega$.

$$\tau = C2 * Rsr = 0.2\mu F * 3\Omega = \underline{0.6\mu s} \tag{6.1}$$

Considering the first current pulse, we notice that the 1/e value of this voltage peak is calculated to approximately 0.94V. A point near this value was chosen to find the decay time of this pulse, and reading out from the graph; $\tau = 2.07\mu s - 1.43\mu s = 0.64\mu s$. For the second current pulse, the 1/e value is approximately 0.98, and the decay time is found to be $\tau = 7.94\mu s - 7.33\mu s = 0.61\mu s$. We notice that the second current pulse is closer to the ideal both in terms of current peak value and time constant. The reason for this might be that the transistor doing the actual conduction of the current in the second pulse, is an n-channel MOSFET, which has faster turn-on time as well as lower on-resistance than the p-channel. We will discuss this further in section 6.2.5 on transistors. Another thing we note is the large current peak of more than 4A through transistor M15 (the blue trace on figure 6.7). This is switching current arising from misaligned n-channel and p-channel timings. Looking at the IRF7317 datasheet, we find that the n-channel has a fast turn-on time (8.1ns) and rise time (17ns) while the p-channel has a slow turn-off time (42ns) and fall time (49ns). The result is a relatively long period of time (91ns - 25ns = 66ns) where both transistor channels are fully or partially open and current flows from 3V to GND.

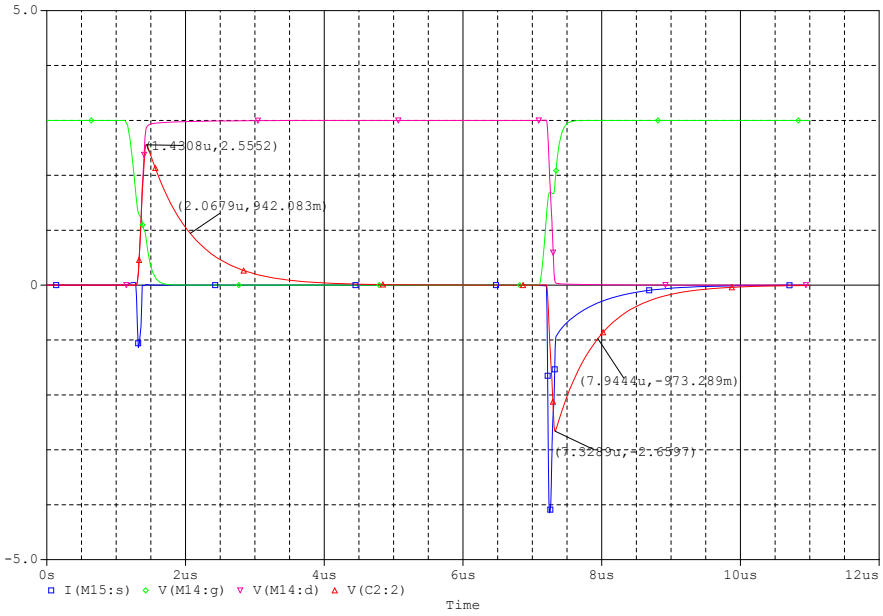

Figure 6.7: Simulation of the Set Reset Totem Pole circuit from figure 6.6. The colours of the graphs correspond to the I and V measurement probes placed on the PSpice schematic in figure 6.6. The peak values of the current pulses are marked for both current pulses, along with the time stamp, as well as the approximate 1/e-values.

6.2.4 Breadboard setup

Some SR circuit prototypes were tested on a breadboard connected to the Altera MAX II development board.

A block diagram of the VHDL code used to test these setups is shown in figure 6.8. The complete code is reproduced in appendix B. The CRU component takes the main 66Mhz clock from the board as input. Internally it divides the clock by 20 to get a clock frequency of 3.30 Mhz, similar to the ICI-3 SCLK frequency. It also contains a powerup state machine which produces an asynchronous reset signal used to initialize all registers. The Reset Timer is a simple counter that drives the maxcount signal high for one clock signal when it reaches its max count. It is used to initiate an SR cycle. The *MAG_SR_Logic* unit drives the actual Vsr signal, which is used to drive the gate of the first transistor stage as seen in figure 6.6. It is a customizable component that takes a generic signal to adjust the number of clock cycles between the Set and Reset pulses. The LED blocks were used for switching the LEDs with a constant frequency for the purpose of visual feedback of a running system. The clk_out signal was simply used to verify the clock frequency on the oscilloscope. The demodulation signals were not used for the SR test, more on those signals in section 6.3.

The significance of on-resistance

When we have a VCC of 3V to work with, and want to achieve a peak current of $I_{max} = 0.8A$ through the SR strap which has a resistance of typ. $R_{SR} = 2.5\Omega$, it

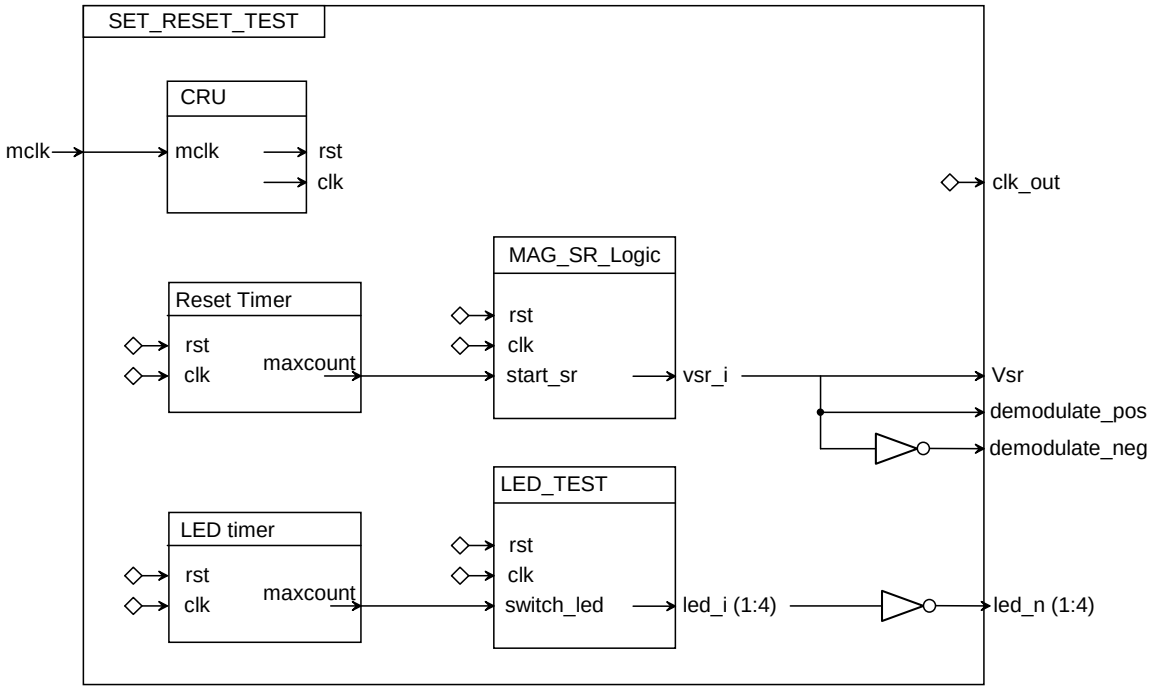

Figure 6.8: The block diagram shows the main building blocks of the SR test code. The Reset Timer determines the timing of the Set and Reset pulses. The Vsr signal is connected to the gate of the first cMOS inverter stage. The demodulation signals are used for the Switching Feedback circuit tested in section 6.3.

becomes clear that the on-resistance of the transistor will be important. Equation 6.2 shows that the theoretical max current through the SR strap (for a typical component) will be $I_{max} = 1.2A$.

$$I_{max} = \frac{VCC}{R_{SR}} = \frac{3V}{2.5\Omega} = 1.2A \tag{6.2}$$

This is a theoretical maximum, assuming we have an infinitely steep flank. The slow discharge from capacitors C1 and C2 (figure 6.5) during the opening of the transistors will ensure that the curve will not be that steep and the peak will be smeared out, resulting in a lower maximum current value. There is not much leeway, and the on-resistance of the transistors must therefore be as low as possible. The turn-on and turn-off times of the transistors are also of importance as they determine the flank of the current pulse. A compromise must be made between short delays and low on-resistance, as they are normally mutually exclusive.

6.2.5 HEXFET power transistors

In the suggested totem pole circuit, Honeywell uses the IRF7509, which is an International Rectifier HEXFET[3] power transistor. It contains a pMOS and an nMOS transistor in a single package. By inspecting the datasheets of several similar transistors, some were selected for testing, including the IRF5851, IRF7317 and IRF7338. These are all pMOS/nMOS combination packages.

IRF7317

The IRF7317 is a power-horse transistor in an SOIC-8 package, with a specified $R_{DS(on)}$ of only $29m\Omega$ for the nMOS and $58m\Omega$ for the pMOS. It is rated to channel more than $20A$ of current with only 3V of gate-to-source voltage and a low drain-to source voltage of less than 1V (more than $20A$ for the pMOS). However, it has a relatively large total gate charge of typically of $18nC$ and $19nC$ (at $V_{GS} = 4.5V$ for n- and p-channel respectively. The MAX II I/O ports can deliver a maximum current of $25mA$. We can make an estimate of the order of the time it takes the CPLD to fill the gate using equation 6.4 and 6.3 [27].

$$t_n = Q_G/I = \frac{18nC}{25mC/s} \approx 0.72uS \tag{6.3}$$

$$t_p = Q_G/I = \frac{19nC}{25mC/s} \approx 0.76uS \tag{6.4}$$

This makes for a rough estimate of the time it takes to turn the transistor on completely. We must take into account that the gate charge is stated for $V_{GS} = 4.5V$. The graphs on figure 6.9 and 6.10 show that at $V_{GS} = 3V$ the on-resistance will be below $15m\Omega$ for the n-channel and below $65m\Omega$ for the p-channel, both given for high drain currents $I_D > 5A$.

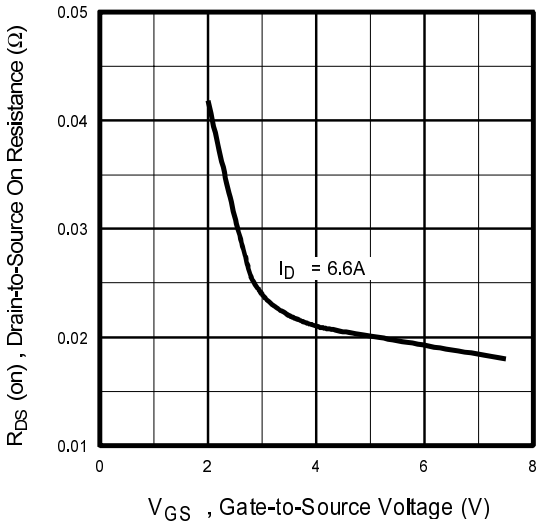

Figure 6.9: IRF7317 n-channel on-resistance, from [26]. As the gate-to-source voltage reaches 2V, the on-resistance drops rapidly from around $40m\Omega$ to below $25m\Omega$ at 2.5V gate-to-source voltage.

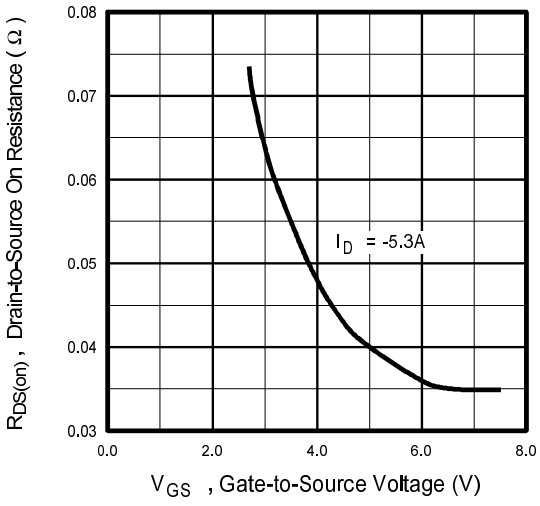

Figure 6.10: IRF7317 p-channel on-resistance, from [26]. The p-channel has a higher resistance than the n-channel, and stays above $60m\Omega$ even at 3V gate-to-source voltage. It will prove a bit more tricky to get the p-channel gate down to a low resistance quickly enough.

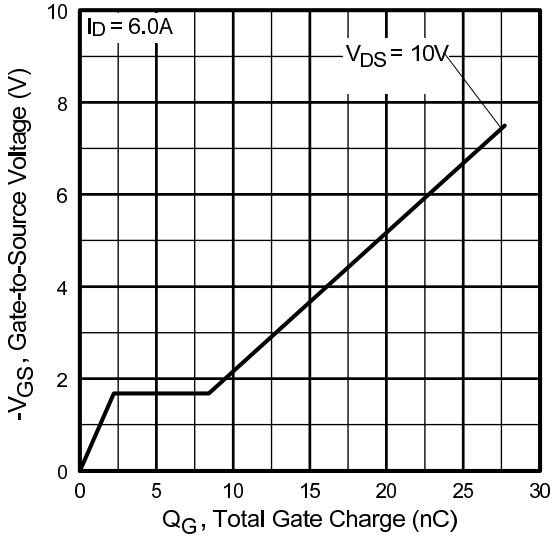

Figure 6.11: IRF7317 n-channel gate charge, from [26]. The total gate charge graph shows how the gate voltage flats out at just below 2V gate-to-source voltage. This is because the gate is absorbing all the charge that is fed into it. During this stage it is important to fill the gate as quickly as possible to avoid a "half-open" state.

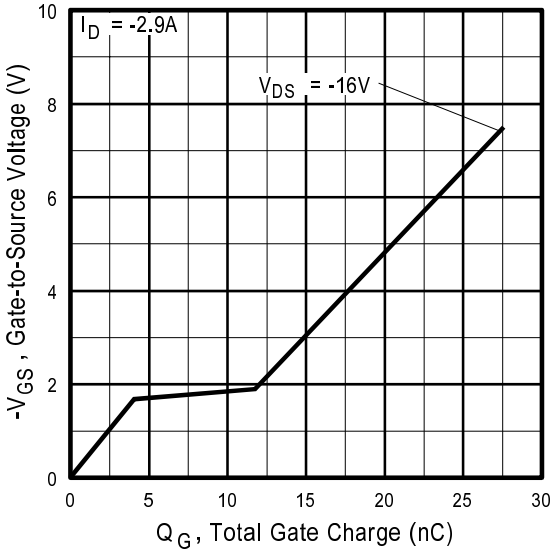

Figure 6.12: IRF7317 p-channel gate charge, from [26]. We see the same tendency for the p-channel as for the n-channel, only it takes even longer to fill the gate. Here it is even more crucial to fill the gate quickly as the gate charge required for low on-resistance is larger.

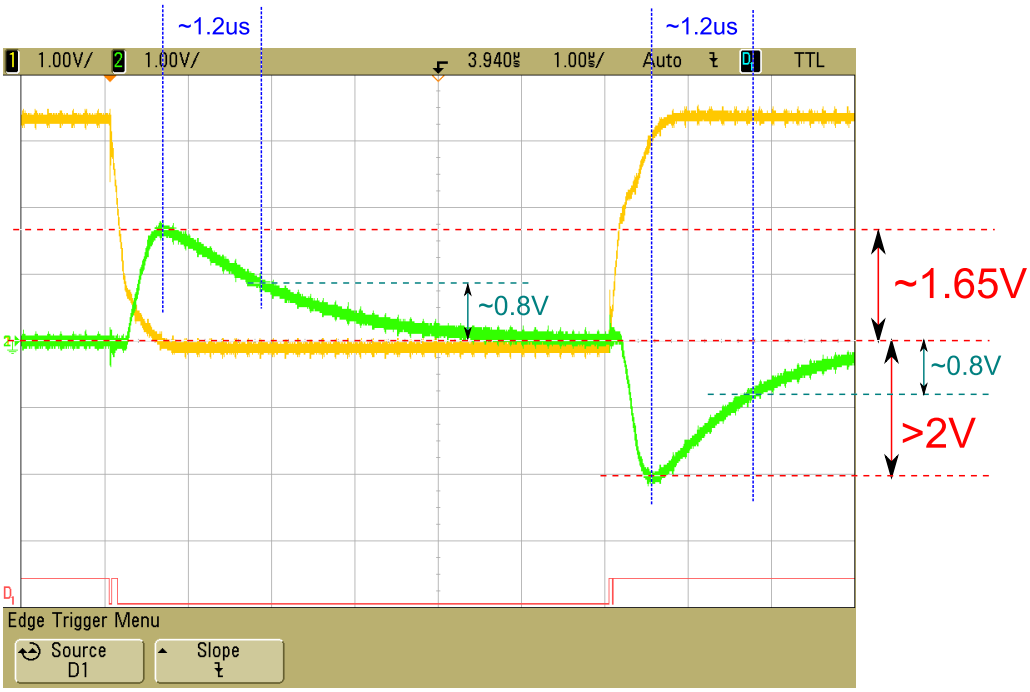

Figure 6.13: Oscilloscope measurement of a SR current pulse using the IRF7317 with the inverter gates driven directly from the CPLD I/O port. Yellow is gate voltage. Green is the voltage over the SR strap model. Bottom red is the digital interpretation of the gate voltage. We can see the ripple as the CPLD tries to drive the gate of this monster transistor.

Figures 6.11 and 6.12 show that the typical gate charge for $V_{GS} = 3V$ is still in the neighbourhood of 15nC for p-channel and 13nC for n-channel. Substituting these charge numbers into equations 6.3 and 6.4 give us equations 6.5 and 6.6.

$$t_n = Q_G/I = \frac{13nC}{25mC/s} \approx 520ns \tag{6.5}$$

$$t_p = Q_G/I = \frac{15nC}{25mC/s} \approx 600ns \tag{6.6}$$

Even with these modified timings, the point is made that it is worth investigating if the I/O port itself can turn on the transistors quickly enough to produce the steep voltage flank needed. Figure 6.13 shows an attempt to perform a set-reset pulse with a single IRF7317. The gates are driven directly from a CPLD I/O port through a 94Ω series resistor. This gives a max output current of $\approx 36mA$ which is more than the recommended maximum value, although it will be just a short current peak of less than $1\mu s$. As the SR strap has a mostly resistive impedance, the strap is modelled with a resistor $R_{SR} = 3\Omega$. We see that the first pulse has a peak of about $1.65V/3\Omega \approx 0.55A$. The second one has a peak of more than $2V/3\Omega = 0.67A$.

[3]HEXFET refers to the shapes used in the mesh structure of the source metalizations on the die. The mesh structure has a hexagonal shape to minimize die size and maximize the channel density.

The capacitance in series with R_{SR} is $0.4uF$, which is the highest value used in these tests, but it should give an idea of the maximum attainable peak value. The yellow graph is the voltage at the gates of the IRF7317, while the green graph is the voltage across R_{SR}. As the load is resistive, the voltage is proportional to the current. We can take note of several points on these graphs. The transistors start turning on at around $V_{GS} = -2V$ for n-channel and $V_{GS} = 2V$ for p-channel. We notice that the first flank is less steep than the second flank. This is because the first current pulse is flowing through the p-channel and the second one through the n-channel. Substituting the new capacitance value into the equation 6.1 for the time constant we get an expected time constant of

$$\tau = C * R_{SR} = 0.4\mu F * 3\Omega = \underline{1.2\mu s} \tag{6.7}$$

This means we would expect the voltage to reach $1/e \approx 1/2.718 \approx 36.8\%$ of its peak value after $1.2\mu s$. We see however that the peak of the first pulse is smeared out, and the 1/e value $36.8\% * 1.65V = 607mV$ is reached only later. On the second pulse however, it is easier to recognize the time constant, as 2V has fallen to about 0.8V after approximately $1.2\mu s$, which constitutes a fall to 40% of the original value. This is not surprising, considering the n-channel transistor delivers the current for this pulse, which means it turns on faster as well as provides a lower on-resistance when turned on. As we saw earlier both the gate charge and the on resistance of the p-channel is larger, so this is a natural effect.

IRF5851

The IRF5851 is a lighter transistor in an MSOP-8 package. At $V_{GS} = 3V$ the on-resistance is $85m\Omega$ for the n-channel ($I_D = 2.7A$) and about $170m\Omega$ for the p-channel ($I_D = -2.2A$). The total gate charge is about $2.5nC$ for the n-channel (at $V_{DS} = 10V$) and $2.5nC$ for the p-channel (at $V_{DS} = -10V$). Comparing the IRF5851 to the IRF7317 uncovers that the latter is far easier to drive. The measurements on figure 6.14 were taken with the IRF5851. The setup was the same as for the IRF7317, except that the series resistor used was 90Ω instead of 94Ω, but the difference is negligible[4]. The figure shows a noticable difference in the slope and the max value of the current pulse, in both the first and the second pulse. This result suggests that the gate of the IRF7317 is not driven with enough current.

IRF7338

Lastly, we will take a look at the IRF7338. It falls in between the IRF5851 and the IRF7317 in terms of both gate charge and on-resistance. At $V_{GS} = 3V$ it has an on-resistance of about $45m\Omega$ for the n-channel ($I_D = 6.3A$) and $100m\Omega$ for the p-channel ($I_D = 3.0A$). The total gate charge is about $4.5nC$ at the n-channel (for $V_{DS} = 6V$) and about $2.5nC$ at the p-channel (for $V_{DS} = 12V$). This transistor was tested in

[4]This is backed up by analysis of other measurements made with larger resistors ($> 200\Omega$), still showing improvements over the IRF7317, but this value was the closest to 94Ω on record

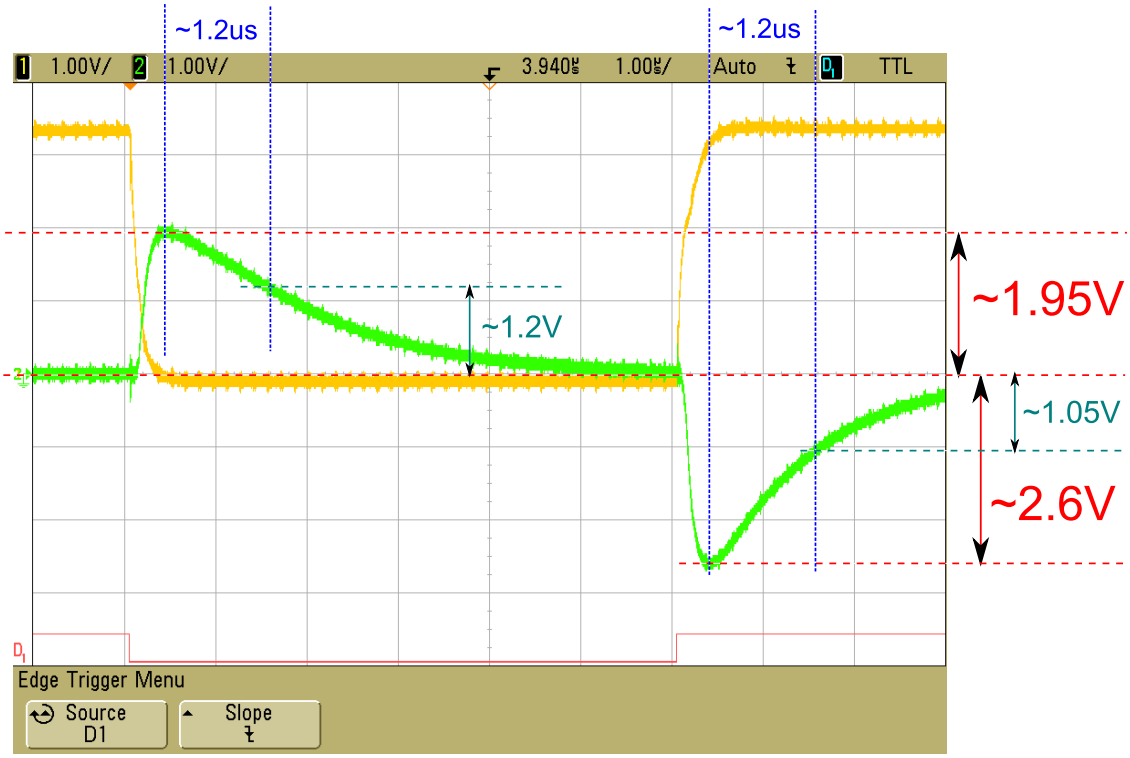

Figure 6.14: IRF5851 SR current pulse. This transistor is easier to drive than the IRF7317 which is reflected in the peak values for both channels. Again the p-channel smears out the peak of the first pulse, by not dropping its on-resistance fast enough. Values are marked as on figure 6.13 for comparison. Bottom red is the digital interpretation of the gate voltage.

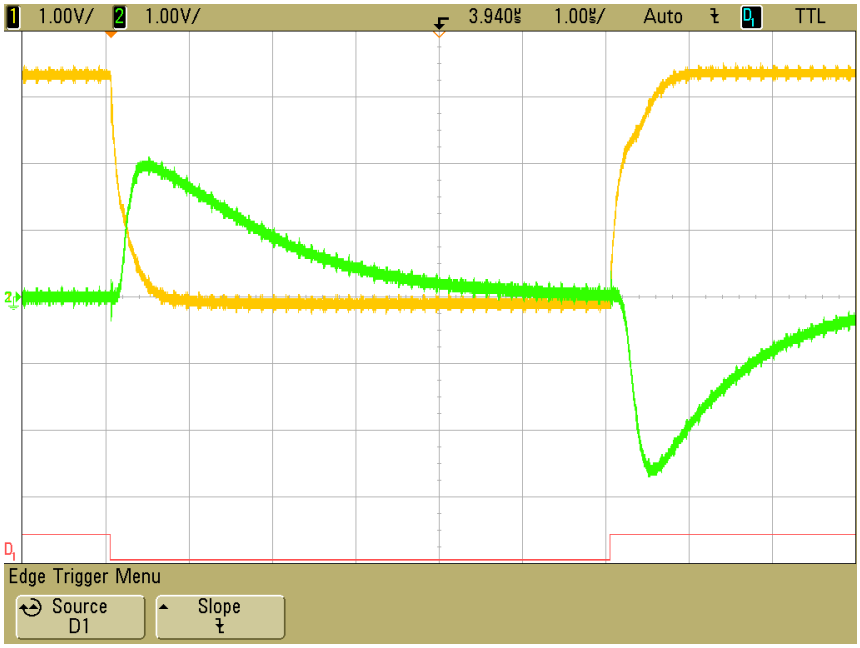

Figure 6.15: IRF7338 SR current pulse. Gate resistor is 120Ω. Yellow is gate voltage, green is the voltage over R_{SR}. Results place in between the IRF7317 and the IRF5851. Bottom red is the digital interpretation of the gate voltage.

the same setup as the two others, but with a series resistor of 120Ω. The result of this measurement is shown in figure 6.15. Not surprisingly, the IRF7338 places in between the IRF5851 and the IRF7317 in terms of results. The slope of the pulse is steeper than its bigger cousin IRF7317 but somewhat less steep than the IRF5851. We notice that the gate voltage takes longer to reach the low state for the first pulse and similarly to reach the high state for the second pulse. These regions correspond to the flat regions we see on the gate charge graphs in figures 6.11 and 6.12 for the first and second pulse respectively. The gate charge is building up while the gate voltage stays largely unchanged. During this time the on-resistance has already dropped and the current has started flowing, so the key to a nice sharp current pulse is to increase that gate charge (i.e. decrease that on-resistance) as quickly as possible.

The reader might wonder if there would be something to gain from lowering the series resistance and simply push the I/O port to the limit. Figure 6.16 illustrates the tendency of the signal to become rippled by lowering the series resistance (a 12Ω resistor was used for this measurement).

6.2.6 The two-stage switcher

To take advantage of the low on-resistance of the IRF7317, ways to buffer the CPLD I/O output were explored. At an early stage, an attempt was made at using a Bipolar Junction Transistor to amplify the I/O current. It would turn out that the result was not optimal, largely because of the need to rely on a strong pull-up resistor for the rising flank.

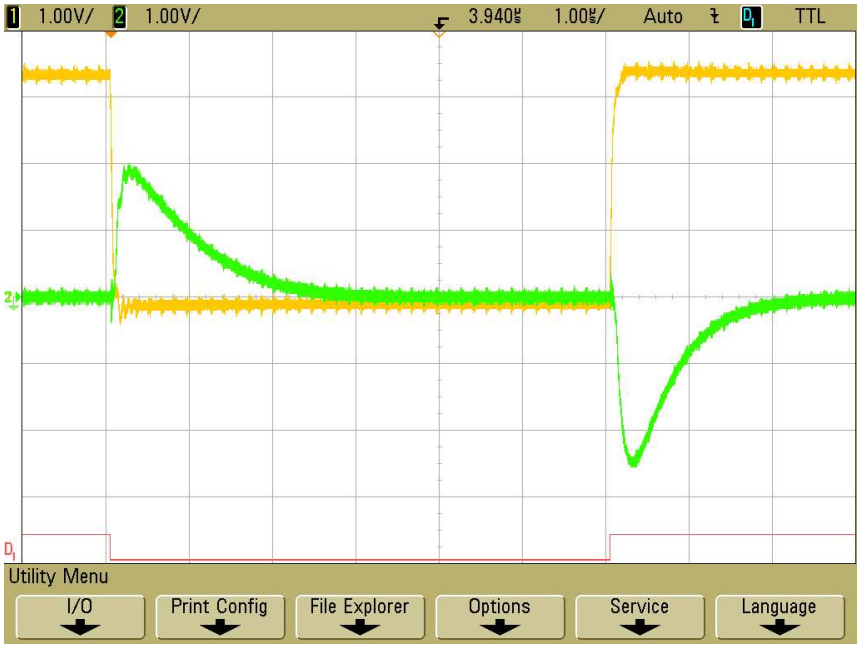

Figure 6.16: IRF7338 SR current pulse Gate resistor is 12Ω. We notice the ripple on the gate voltage (yellow) and the resulting ripple on the output voltage pulse (green). This happens because the CPLD I/O pin cannot stabilize the gate voltage with such low load resistance. (Note: this measurement was taken with a capacitor $C2 = 200nF$ instead of $400nF$ in series with R_{SR}). Bottom red is the digital interpretation of the gate voltage.

It would turn out that another cMOS inverter was better for the first stage, and after testing numerous setups with various values for series transistors, a good setup was found. IRF5851 and IRF7317 were connected as shown on the PSpice schematic in figure 6.6 (except that the IRF7338 on the schematic is replaced by the IRF5851). Figure 6.17 shows the measurements from this setup. Again, the green curve is the voltage on the gate of the IRF7317 inverter. In this measurement an averaging function on the oscilloscope was used, hence the noise-less lines. We can see that the voltage stays above at least 2.35V for more than 100ns. This amounts to a current value of $I_{SR} = 0.94A$ with a typical SR resistance of 2.5Ω. If we use the worst-case value for the resistance of $R_{SR} = 3\Omega$, a current peak value of $I_{SR} = 0.78A$ is found, which is just below the stated minimum. Thus, as long as we do not encounter non-typical parts with a high Strap Resistance we should be fine. The temperature coefficient of the SR strap is high (max. $4100ppm/°C$), and amounts to about 0.5Ω for a temperature swing of $50°C$. A system experiencing high temperature swings should therefore consider using a different solution for the SR circuit (see chapter 10).

A series capacitance of $C_{series} = 300nF$ was used in this measurement (compared to the $400nF$ seen in the previous section), which leads a shorter duration of the Set and Reset pulses. We notice the time constant of $\tau = 0.3\mu F * 3 = 0.9\mu s$ and check the graph for adherance to the same pattern of 1/e decay after a time of τ. The first pulse decays to approximately 33% within time τ, which is surprisingly fast. This might be because the actual (ideal) peak comes sooner than measured. The second

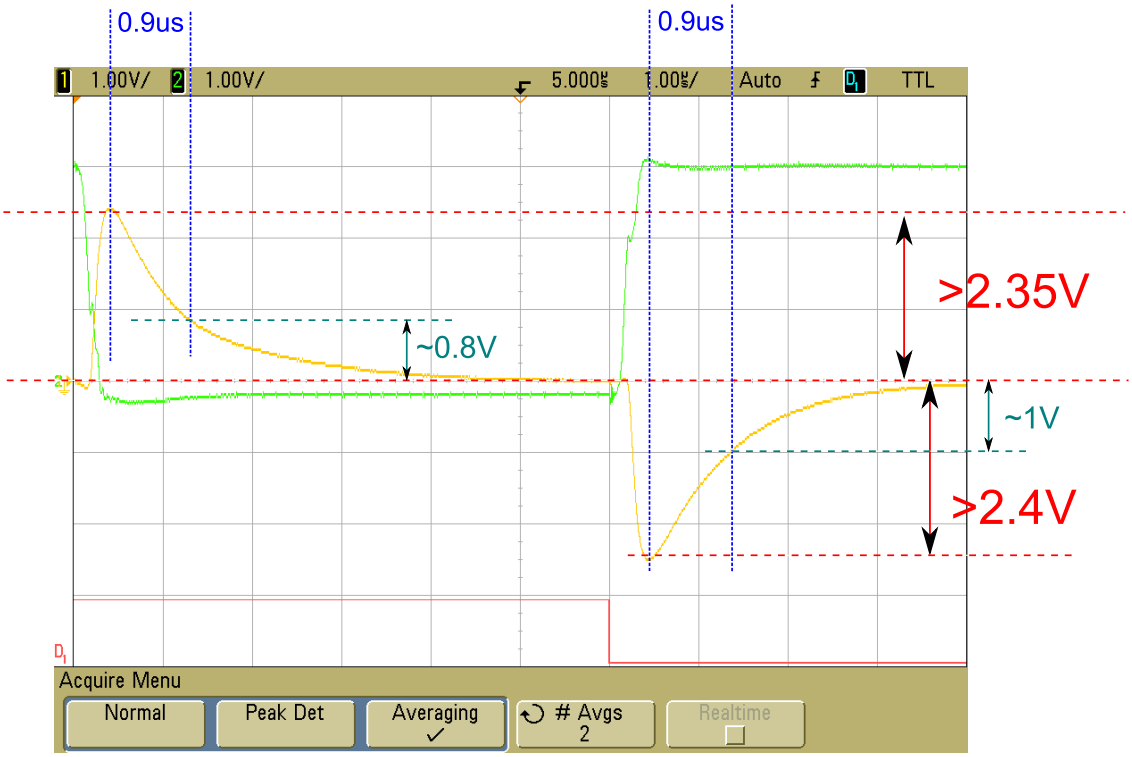

Figure 6.17: IRF5851 and IRF7317 SR current pulse. This measurement uses a capacitor value of $C2 = 300nF$ in series with R_{SR}. Green is the gate of the IRF7317 inverter. Yellow is the voltage across R_{SR}. When we measure slightly below the peak, we find a value that is kept for at least 100ns (one tenth of a grid square).

pulse decays to approximately 40% after τ, which more in tune with the expected figure.

6.3 Switching feedback circuit

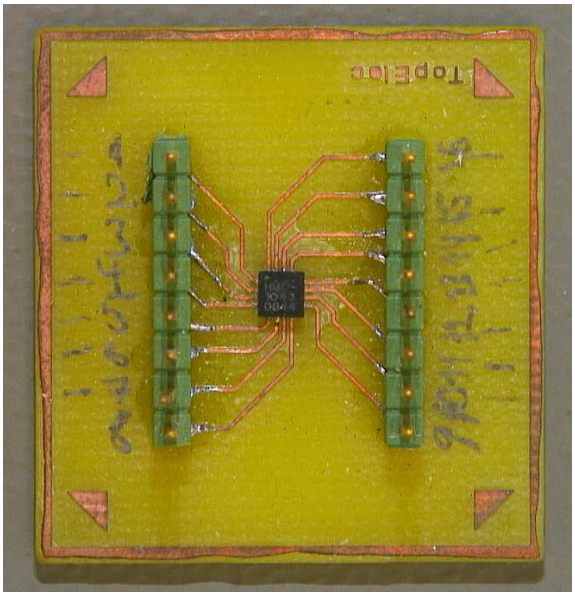

Figure 6.18: A breakout board with an HMC1043 magnetometer (Thanks to the Microelectronics Lab at the Institute of Informatics, for help with producing this small pcb.) The pad spacing is 0.5mm, and some soldering rework was required to achieve proper connection for all the pads.

For this test, a breakout board was made for the HMC1043 magnetometer (see figure 6.18). This was placed in a breadboard configuration which was connected to the Altera MAX II development board (see figure 6.1). The code used during these tests is identical to the code used for the SR circuit test in section 6.2.

As discussed in section 4.6.5 on page 44 the Switching Feedback Circuit will take advantage of the Set and Reset modes of the magnetometer to get opposite polarity measurements. This output will be integrated and applied as a bias to the input of the first amplifier stage. To assess the hardware requirements of this circuit there are several elements that need to be considered. Keeping in mind figure 4.11 on page 46 we will discuss these elements in the coming sections.

6.3.1 PSpice: Switching Feedback

Figure 6.19 shows the switching feedback circuit implemented in a PSpice schematic. It does not include the demodulation stage, which was simulated separately. To simulate the square pulse output from the magnetometer, inputs V+ (V34) and V- (V23) were stimulated as shown in figure 6.20. The values in the figure are given in mV. The stimulation was set up to introduce an offset. We see that Vdiff cycles

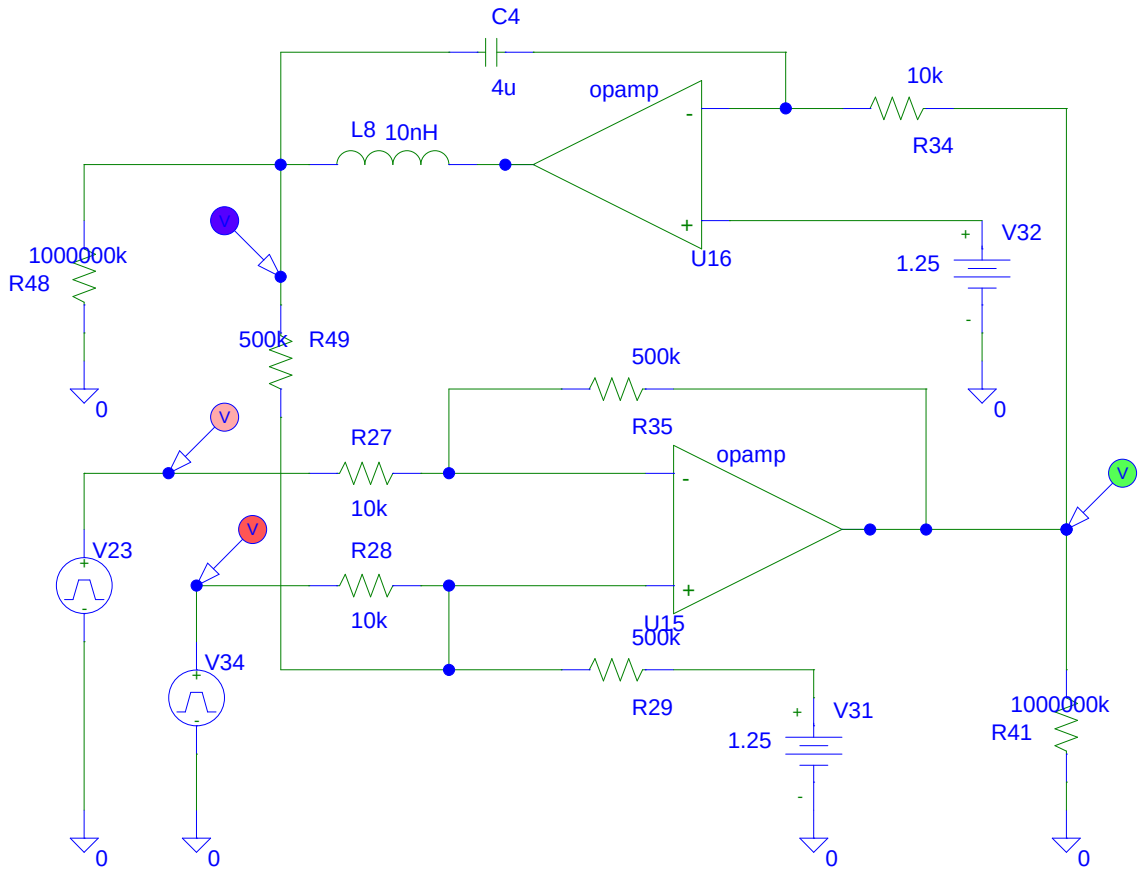

Figure 6.19: PSpice Switching Feedback schematic. This is essentially the same circuit shown in figure 4.11 in section 4.6.5, but without the demodulation stage. See that section for a description of the circuit principle. V23 and V34 are the stimulation voltages used to model the magnetometer output. Resistors R48 and R41 as well as inductor L8 were connected to allow simulation.

from +6 to -2, so it is actually a square pulse of amplitude $\pm 4mV$. Amplified by a factor of 50, we get a square pulse of $\pm 4 * 50 = \pm 200$. As the opamp is connected as a differential amplifier with a reference of $V_{REF} = 1.25V$ we expect the output to swing around V_{REF} with an amplitude of 200, from 1.050V to 1.450V. As we see on figure 6.22 the simulated output values are 1.448V and 1.052V, which means that we are pretty close to the expected output. This result can be improved further by increasing the time constant, which means that the output settles slower. This leads to long simulation times and large datasets to handle. Another thing that can be changed is the feedback bias resistor R49. Giving it a higher value will increase the accuracy of the results, but it leads to bias voltages (at the output of the integrator) closer to the border of operation (0V or 3V, depending or polarity), so that should be used with caution. In figure 6.21 we can see that voltage represented as blue squares.

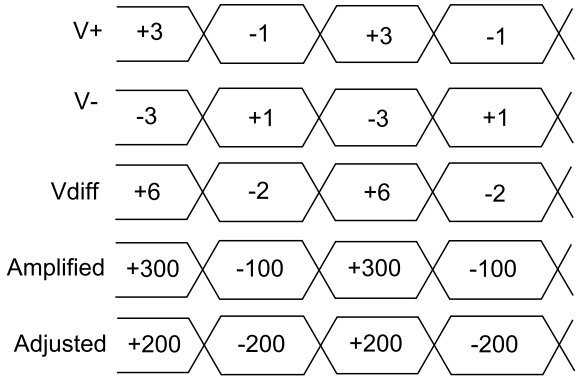

Figure 6.20: Stimulation for the Switching Feedback simulation. Figures are given in mV, and the amplified values are relative to $V_{REF} = 1250mV$ because of the differential amplifier setup. The amplification in the circuit was 50.

6.3.2 PSpice: Demodulation

The demodulation circuit was simulated separately using the circuit shown in figure 6.23. The two switches open and close once, so we simulate one cycle of the demodulation. The stimulation for the simulation is a square pulse V1 switching between 750mV and 1750mV with a pulse width of of $175\mu s$, starting by going to 1750mV (or 1250mV + 500mV). At the start, U8 is open and U7 is closed. That means only V1 is present on both inputs of the opamp, and the opamp is working as a voltage follower. At this point 1750mV is present both on the output and the input. After $175\mu s$ the input voltage switches to 750mV (or 1250mV - 500mV). At the same time, the switch U7 closes, applying 1250mV on the non-inverting input of U1. At this point the amplifier is working as an inverter, inverting the input around 1250mV, leading to an ouput of $1250 + 500mV = 1750mV$. The simulation results are shown on figure 6.24.

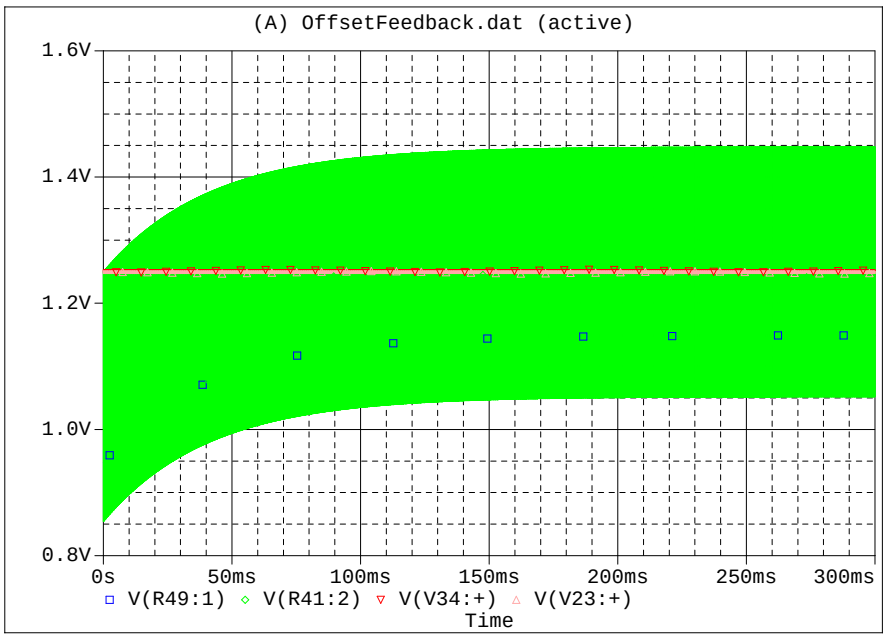

Figure 6.21: PSpice Switching Feedback simulation. Green is the switching output voltage. Blue squares represent the bias voltage at the output of the integrator. The red and orange traces are the modelled non-amplified magnetometer output voltages.

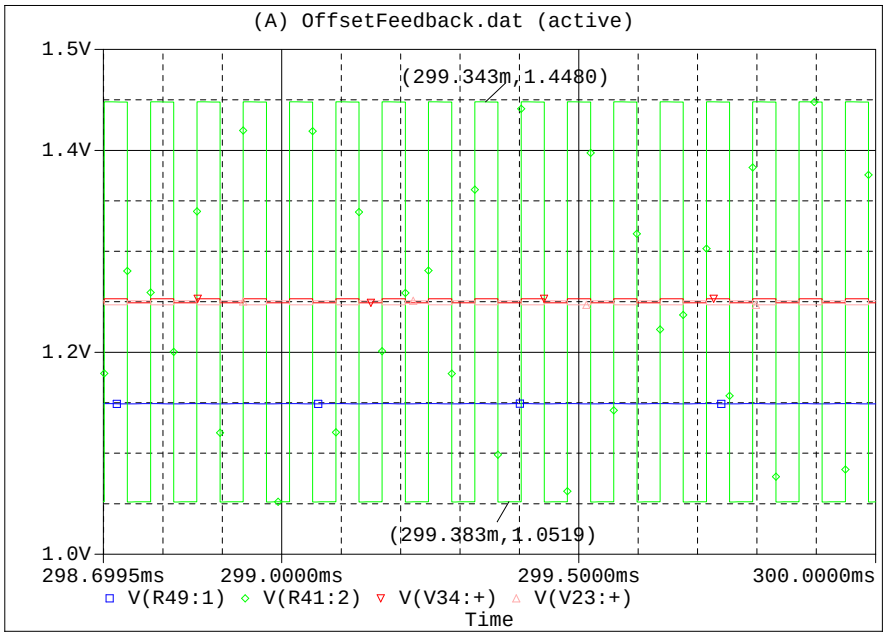

Figure 6.22: PSpice Switching Feedback simulation zoomed in on the area after the ouput has settled. We take note of the positive polarity value of 1.448V and the negative polarity value of 1.052V.

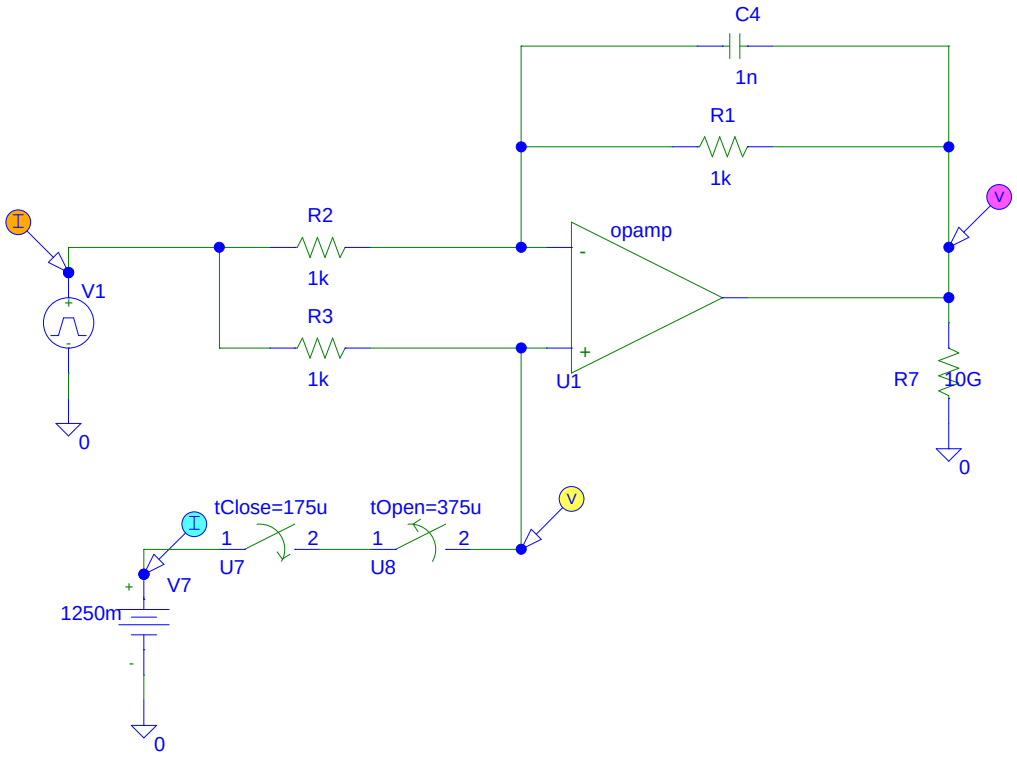

Figure 6.23: PSpice Demodulaction schematic. Based on the demodulation stage of the switching feedback circuit in figure 4.11 in section 4.6.5.

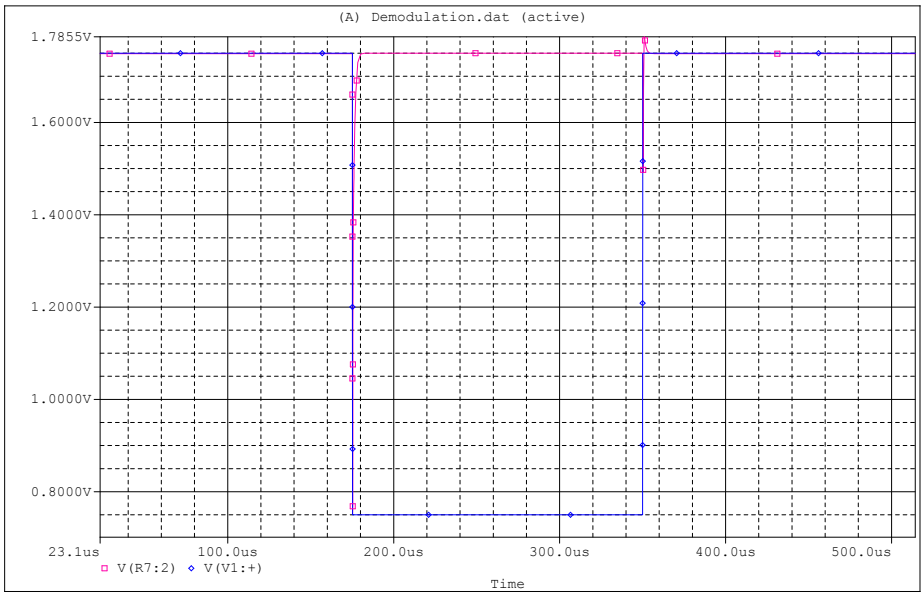

Figure 6.24: PSpice Demodulaction schematic

6.3.3 Raw Output

In section 4.3 it was determined that the minimum voltage step of the raw magnetometer output will be around $300nV$. If a worst-case estimate of noise is taken into consideration (see section 4.4), the smallest measurable voltage will be in the range of $1.6\mu V$. The true attainable resolution most likely lies somewhere in between these numbers. The fact remains that we have the challenge of measuring a voltage difference in the order of $1\mu V$.

6.3.4 Opamp selection

There are two crucial properties that are important for the selection of the Operational Amplifiers to use in this circuit, Input Offset Voltage and Slew Rate.

Input Offset Voltage

An ideal operational amplifier setup with a gain of 1, will have an output exactly the same as the input. In reality, there will be a small difference between the input and the output voltages, and that difference is the same as the input offset voltage. With higher gains, the output offset voltage scales with the gain. For our application, where the voltage to be measured can be as low as $< 1\mu V$, the input offset voltage becomes an important property of the opamp.

Slew Rate and GBP

The output from the differential amplifier seen in figure 6.19 should have a square pulse output with a frequency of about 2.9kHz. For optimal functionality of the demodulation circuit, the square pulse should have flanks as steep as possible, to ensure that the switching happens fast enough for the demodulation output to stay smooth. Figure 6.24 shows that with an ideal opamp and a rise and fall time of 100ns on the input, the demodulated signal is not a perfectly smooth output, but has some peaks around the switching areas. The slew rate of the operational amplifier says how fast the amplifier is able to change its output voltage level, and is therefore an important property. The slew rate is usually given for a gain of 1, and get significantly worse for higher gains. As we will use higher gains, another important property to consider is the Gain Bandwidth Product (GBP), which is an indicator of how fast the opamp can work on higher gains. An opamp with a high slew rate as well as a high GBP will be needed.

OPAx376

Finding a suitable opamp for the application was not a trivial task, as the desired properties of low input offset voltage and high GBP and Slew rate are somewhat contradictory. After searching the market thouroughly, and doing some laboratory testing, an opamp was selected, the Burr-Brown OPA376/2376/4376 (1/2/4-channel). It has an input offset voltage of typically $5\mu s$, a slew rate of $2.2V/\mu s$ and a GBP of

5.5Mhz. With a relatively low quiescent current[5] typically $760\mu A$ and a maximum output current of $10mA$, this is a well-rounded all-purpose opamp that meets our needs. This magnificent work of silicon will be used in all breadboard tests and forms the foundation of the analog part of the final design.

6.3.5 Breadboard measurements

To test the principle behind the Switching Feedback Circuit with real components, a breadboard setup was created similar to that of the schematic in figure 6.19, but this time, the OUT+ and OUT- outputs from a real HMC1043 magnetometer are connected in place of voltage sources V23 and V34. The bridge voltage connected across VB and VSS of the magnetometer is 2.5V and the Voltage reference is equal to half the bridge voltage; $V_{REF} = 1.25V$. In the first example we will be looking at, R27 and R28 have values of $5k\Omega$ while R35 and R29 have values of $620k\Omega$. This gives a total amplifier gain of 124. To investigate the limits of operating frequency, a high SR frequency of 6.4kHz is employed, and the SR charge pump capacitor (C2 in figure 6.6) has a value of 200nF. Regrettably, the integrator time constant values (C4 and R34 in figure 6.19), were not written down at the time of testing, so those values are unknown. The feedback bias resistor R49 is also unknown, although it is assumed to be $620k\Omega$.

Figure 6.25 shows the results of this setup. The green trace shows the SR current, while the yellow trace shows the output voltage. We notice that the swing of the square pulse is approximately $\pm 30mV$. It is noted that a large part of the duty cycle is spent on climbing to the correct voltage. This is by and large due to restrictions in the operational amplifier slew rate[6]. Although the finished system is not expected to be running at such a high frequency, the performance is not good enough to be able to obtain a smooth output from the demodulation stage. A new design was attempted, which would turn out to do the job better, the dual amplifier stage. The principle behind this amplifier is illustrated in figure 6.26. The advantage to such a design is that each amplifier stage is not pushed that hard, and better slew rate is expected.

The first stage is a differential amplifier. In this application, resistors R1 and R3 will be equal, and resistors R2 and R4 will be equal. This means that the amplified output from the first stage is given by equation 6.8.

$$V_{OUT1} - V_{REF} = A1 * (V_{IN+} - V_{IN-}) = (RA2/RA1) * (V_{IN+} - V_{IN+}) \tag{6.8}$$

$$V_{OUT2} - V_{REF} = A2 * (V_{OUT1} - V_{REF}) = (1 + \frac{RB2}{RB1}) * (V_{OUT1} - V_{REF}) \tag{6.9}$$

[5]The word *quiescent* stems from the latin present participle of *quiescere*, which means *to become quiet* or *to rest*. It is a measure of the current drawn by a circuit when not driving any load, i.e. when idle.

[6]Earlier tests with slower operational amplifiers showed an even poorer performance in this regard

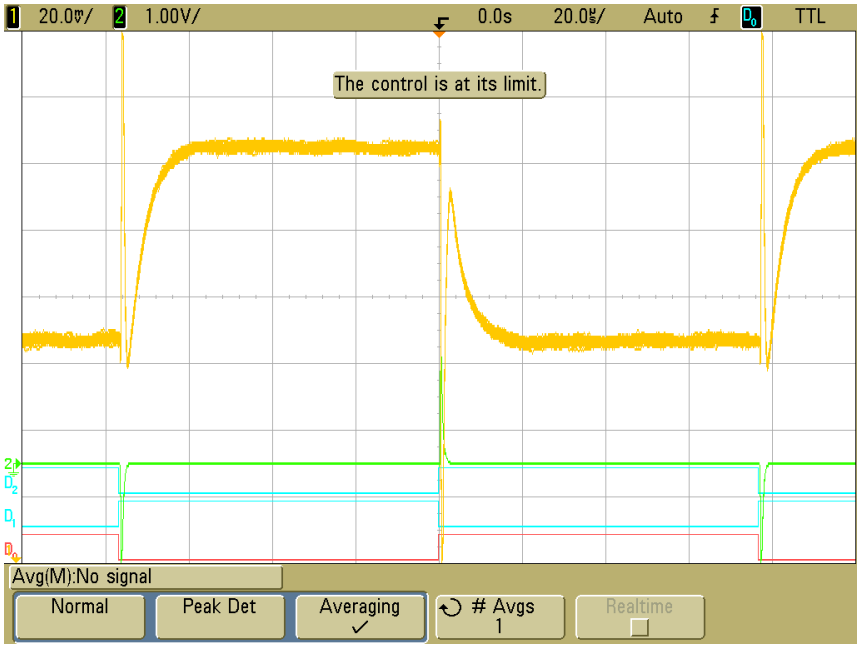

Figure 6.25: Output from a single-stage switching feedback amplifier. Total gain is 124, the SR frequency is 6.4kHz. Green trace is the SR current. Yellow trace is the output voltage. The red trace shows the digital SR control signal.

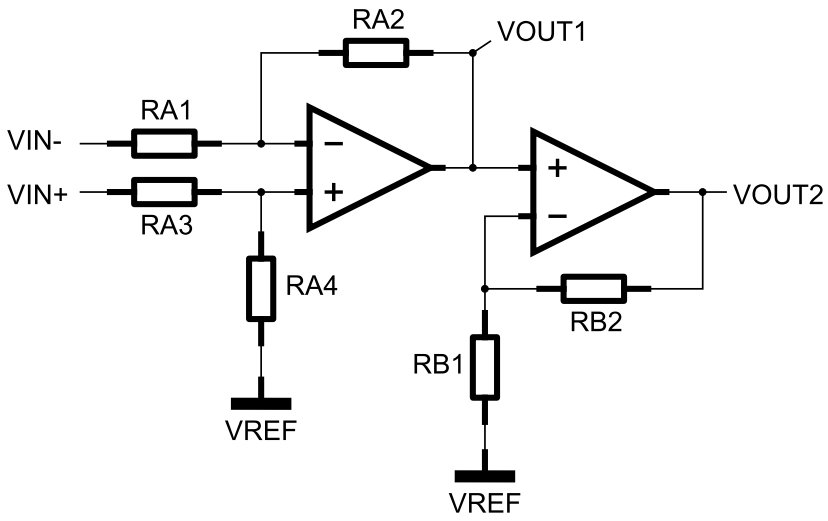

Figure 6.26: Dual stage amplifier principle sketch. The first stage is a differential amplifier, amplifiying the input difference with respect to VREF. The second stage is a common non-inverting amplifier (also using VREF as the reference voltage)

$$V_{OUT2} - V_{REF} = A2 * (V_{OUT1} - V_{REF}) = A2 * A1 * (V_{IN+} - V_{IN-}) \tag{6.10}$$

$$A1 = (RA2/RA1) \tag{6.11}$$

$$A2 = (1 + \frac{RB2}{RB1}) \tag{6.12}$$

The second stage is a non-inverting amplifier which means that its amplified output is given by equation 6.9. To find the total amplification through the system, we combine equations 6.8 and 6.9. Combining the two equations we get equation 6.10 and we see that the total output amplification can be found by multiplying the two separate amplifications. The formulas for the amplification of the two stages are given by equation 6.11 and 6.12.

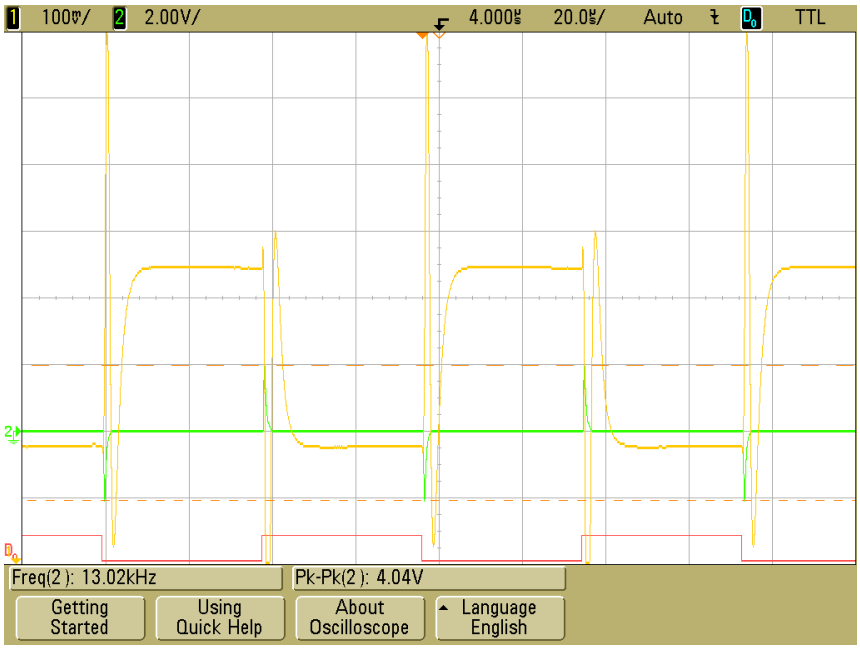

Figure 6.27: Output from a dual-stage switching feedback amplifier. Total gain is 420, the SR frequency is 12.8kHz. Green trace is the SR current. Yellow trace is the output voltage. The red trace shows the digital SR control signal.

A second setup was tested similar to the first one, but this time with a dual-stage amplifier. The resistor values for the first amplifier stage, using the notation from figure 6.26, are $RA1, RA3 = 10k\Omega$ and $RA2, RA4 = 200k\Omega$, resulting in an amplification of $A1 = 20$. The second stage employs the values $RB2 = 200k\Omega$ and $RB1 = 10k\Omega$ giving an amplification of $A2 = 21$ for the second stage. The total amplification of the system is therefore $A_{TOT} = A1 * A2 = 420$. The SR frequency this time around is 12.8kHz. As we see on figure 6.27 two amplifier stages do a much better job of keeping the pace. A total swing of almost $\pm 150mV$ is seen on the

yellow trace, which is the voltage output from the last amplifier stage. Considering the frequency of 12.8 kHz, which is more than 4.4 times faster than the 2.9kHz frequency which will be used in the final design, the dual-stage solution seems up for the challenge.

6.3.6 Demodulation

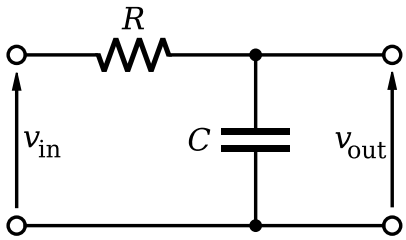

Figure 6.28: Principle sketch of a single-stage RC filter.

Finally, a circuit with a demodulation stage was tested. The setup was similar to the dual-stage amplifier test, and the test was done with an SR frequency of 12.8kHz. The total gain in this test was $A_{TOT} = A1 * A2 = 10 * 11 = 110$, and a simple passive RC filter as seen on figure 6.28 was applied at the output of the demodulation stage. The time constant used for this filter is $\tau = RC = 10k\Omega * 1\mu F = 10ms$, and the cutoff frequency is 100 Hz (by equation 6.13). The output from this setup is shown on figure 6.29.

$$f_c = \frac{1}{2\Pi\tau} = \frac{1}{10ms} = 100Hz \tag{6.13}$$

The VHDL code used for this test is similar to the code used in the previous sections for the SR circuit tests and the Switching Feedback circuit tests, except that the demodulation signals from figure 6.8 are now used. The full code is reproduced in appendix B.

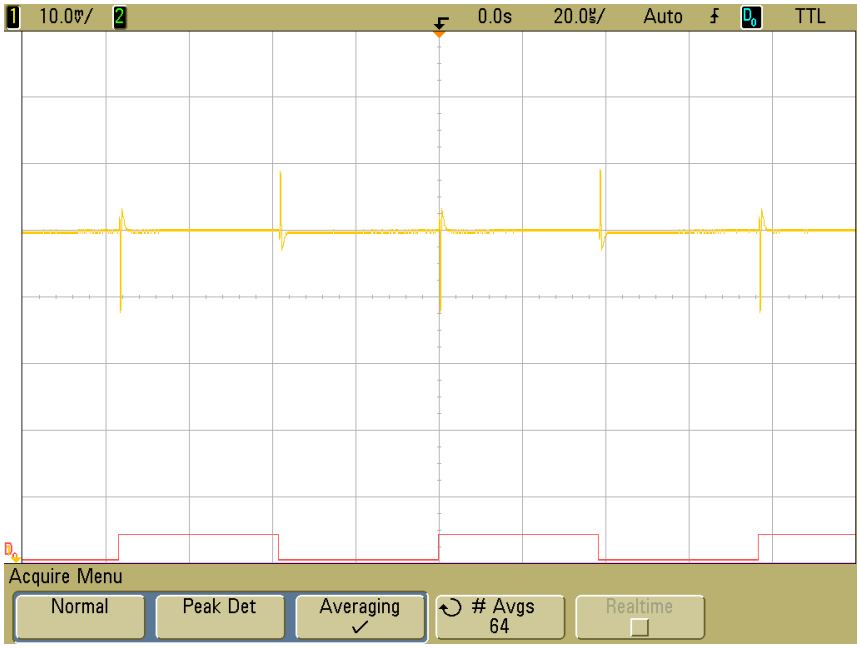

Figure 6.29: Demodulated output from a dual-stage switching feedback amplifier. The signal is run through a single stage passive RC filter with a cutoff frequency of 100Hz. Total amplifier gain is 110 and the SR frequency is 12.8kHz. The red trace shows the digital SR control signal which is synchronized to the demodulation signal. The center of the display represents $V_{REF} = 1.25V$.

Chapter 7

System design

Two hardware designs were made during the course of this work, the draft version (0.5) and the final version (1.1). The final version is based on the draft version, and is largely identical except for a few modifications. Only the final design will be presented here. I will go through the most important aspects of the circuit explaining the design choices and making references to earlier relevant theory sections where applicable. Notes will be made at specific points where the final design differs from the draft design.

7.1 Mechanical Constraints

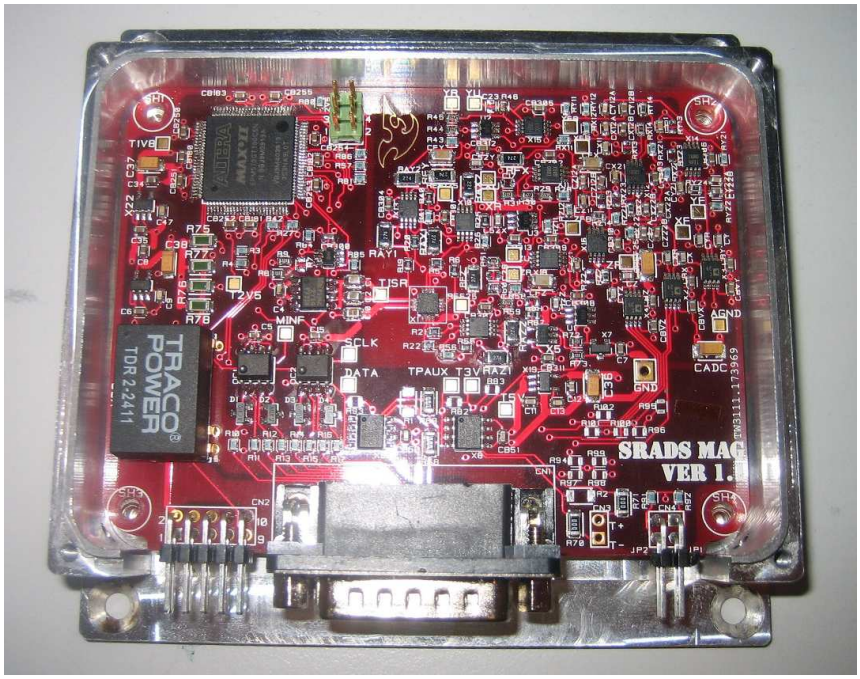

Figure 7.1: The bottom part of the instrument housing, containing one instrument PCB.

The mechanical housing is based on the same box used for the mNLP-instrument detailed in [5]. The same pcb template size is used here, a board measuring $94mm * 71mm$ with a shape to fit the box (see figure 7.1). The board outline can be seen in appendix D. The box is made out of aluminum and has enough room for at least two pcbs stacked on top of eachother.

7.2 Power distribution and grounding

As shown on figure 7.2, the ICI-3 Power Distribution Unit (PDU) provides a 28V and a 0V line to the instrument. The instrument itself is responsible for isolating itself from the PDU. For this design a DC/DC converter from Tracopower was chosen, the TDR2411. This is a 2 Watt DC/DC converter with an input range of 18-36V and a regulated output of 5V. It has an Input/Output isolation of 1500V, which should give good protection against power surges from the PDU. The switching frequency of the unit it 100kHz, which should be sufficiently high to be effectively eliminated by the 100Hz LP filter implemented in section 7.9.

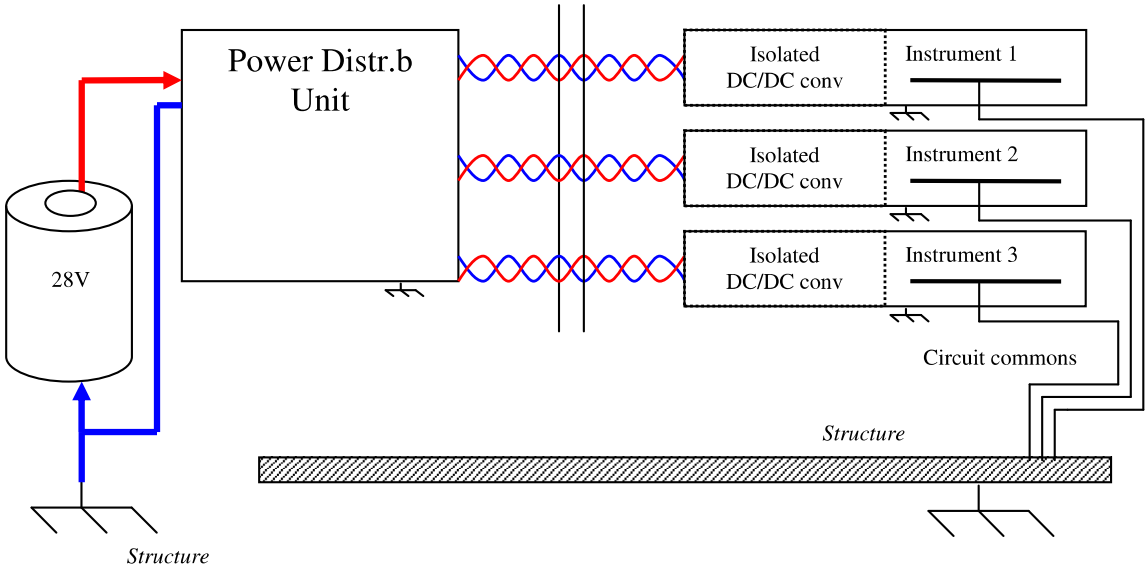

Figure 7.2: Power distribution on the ICI-3 [17]. The Power Distribution Unit delivers 28V and 0V lines to the instrument. On the receiving end sits an isolated DC/DC converter, and the instrument ground is connected to the structure.

7.2.1 Ground Sources

As shown on the Power Block schematic in appendix D, the instrument is designed with three possible ground sources, selectable by zero Ohm resistors. The ground can be shorted to the 0V line from the 28V power supply, which is useful in a lab environment where the power supply has a noise-free 0V line. It can be connected through a test pin at the ground Star Point (see section 7.2.3). This is useful if we

want to connect directly to the metal housing of the istrument through a soldered wire. The last option is to connect ground through Pin6 on the DSUB connector, which was the solution selected for the integration in the rocket, as well as most commonly used in lab tests.

7.2.2 Voltage regions

There are several voltage regions on the instrument (see the Power Block schematic in appendix D). Each of those regions will be explained below.

5V

This is the raw output from the DC/DC converter, and all current through the system is sourced by this. A tantalum capacitor is used at the 5V output from the converter to stabilize the voltage and make sure we have a stable availability of charge. The only component using the 5V power supply directly is the DATA signal transmitter discussed in section 7.3 (although for simplification the same tranceiver was used for UART communication). It is used to make sure that the current for the DATA signal is strong enough to be properly detected by the encoder.

3V

For the rest of the circuit, 3V is used as the primary voltage source. The reason for this design choice was that the CubeSTAR will have a 3V line available for its on-board instruments, and the core design of this instrument takes that into consideration. A Low Drop-Out Linear Regulator (LDO) is used to convert the 5V from the DC/DC to a stable 3V voltage. The 3V voltage region is made into a power plane reaching over most of the pcb. (see Inner VCC Layer in appendix E).

2.5V

This voltage region is employed solely for the I/O supply of the CPLD. The MAX II is specified for either 1.8V, 2.5V or 3.3V I/O operation and, as we cannot guarantee a 3.3V stable supply on the CubeSTAR, the 2.5V I/O standard was chosen. For connection to a total of six pins on the package, a 2.5V plane was designed just around the CPLD (see Inner VCC Layer in appendix E)

1.8V

For the MAX II G and Z components, a separate 1.8V voltage supply is required for the Internal VCore Voltage. This was implemented using the same series of regulators used for the 3V and 2.5V lines, the Texas instruments TPS730xx.

7.2.3 Digital and Analog ground

The layout of the Inner GND Layer in appendix E shows the splitting of the Ground Plane into two sections by a thin slit. This is done to isolate the noise sensitive analogue signals from the noisy switching of the digital components. Care was taken to run all traces near the slit connecting the ground sections, which can be seen when comparing the Bottom Electric and Top Electric layouts to the Inner GND layout. As discussed in [11], this is done to make sure the return paths of all signals follow the same route as the signals, and to avoid current loops. The exception is the SR current trace seen on the Top Electric layout. This is acceptable because the trace is isolated electrically from the other pins of the sensor, and the current goes in and back out the same way. The polarity demodulation digital signal takes a small short cut, which was done as a compromise to avoid an overly long-winded trace. However, this is only one 2.5V digital signal, and the actual current through it is negligible. It was also decided to clear an area around the sensor of GND and VCC planes. To ensure plenty of available 0V charge between the planes, the entrance point from the external ground to the GND plane is at a testpoint at the center of this slit.

7.3 PCM Encoder Interface

As discussed in section 5.1 the instrument is on the receiving side of four control signals from the PCM encoder. As shown in figure 7.3, those signals are distributed by differential lines; one positive and one return line for each signal. The instrument is responsible for isolating itself from those signals, and optocouplers are normally employed for this task. This design uses the HCPL-063L 3.3V optocouplers which are specified for 15MBd communication which is enough for the 3.33Mhz frequency of the SCLK signal. For digital communication with the CPLD, they employ an open-drain architecture, which means that the logical high level is decided by the pull-up voltage. $1k\Omega$ resistors were used for pull-up for a compromise between signal rise times and current drain (A pull-up to 3V through a $1k\Omega$ resistor, give a current drain of 3mA for each signal while it is sourced low by the optocoupler.)

7.4 Programmable Logic

Taking a look at figure 7.4 we see how the design is built up. The *CRU_CONTROL* unit is the main control unit in the design. It instantiates the internal oscillator, which gives a master clock of 3.3-5.5Mhz. This was done to give the instrument the ability to function without the serial clock from the ICI-3 encoder. The *ADC_interface* controls ADC communication which is described in section 7.5. The shift registers are controlled in a similar fashion to what we saw in section 6.1. The *PCM_interface* is detailed in section 7.3.

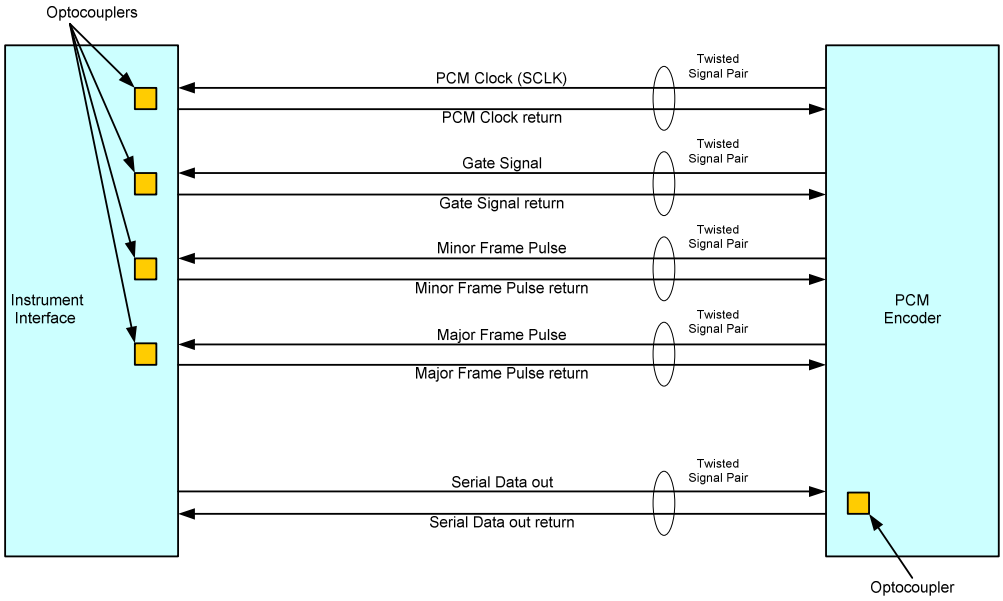

Figure 7.3: PCM encoder communication lines. The incoming signals from the encoder must be isolated locally by the use of optocouplers. The DATA signal is driven by the instrument with a standard RS422-driver, and is isolated at the encoder side.

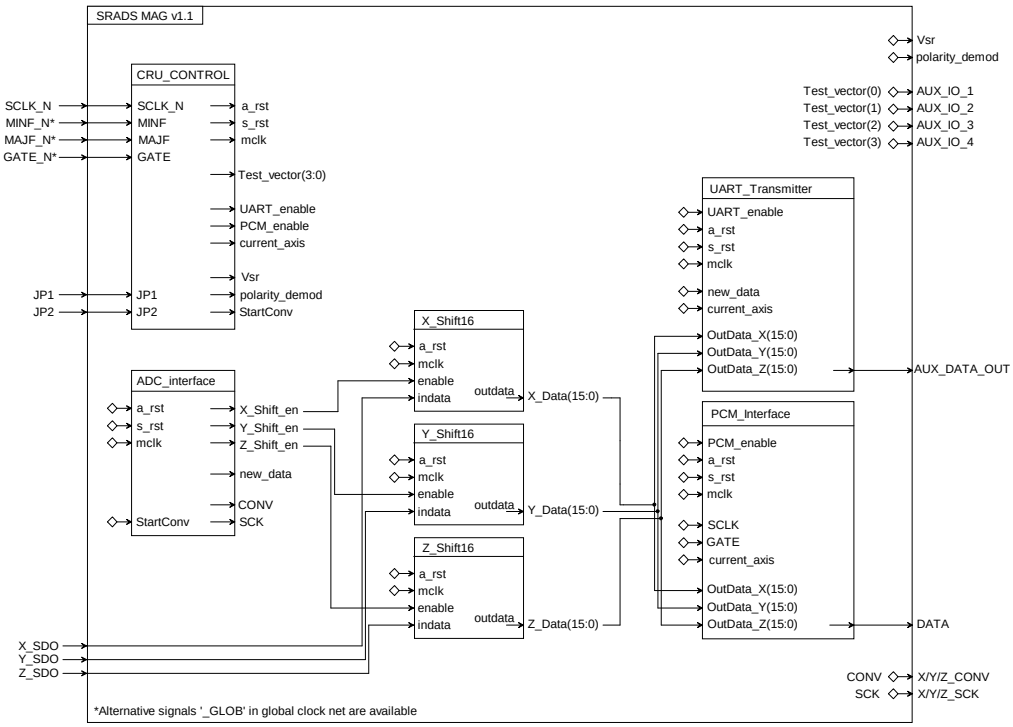

Figure 7.4: Top Block. Using the AD7684 ADC, *DCLOCK* is used to drive the shift registers.

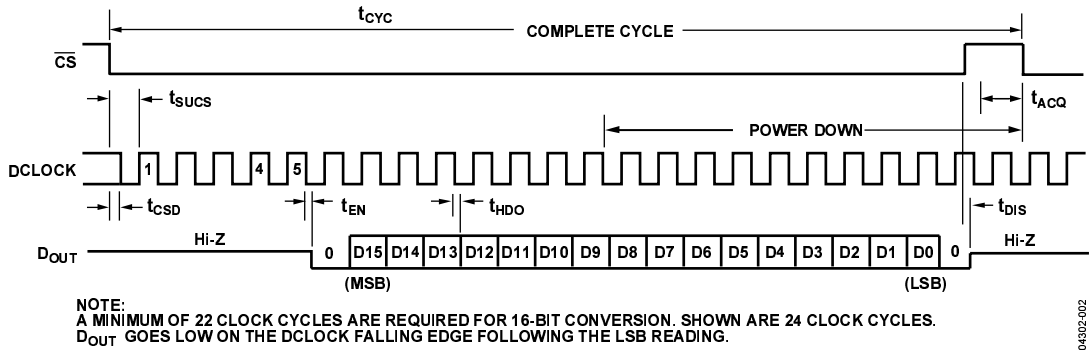

Figure 7.5: Timing diagram for communication with the AD7684, taken from [16].

7.5 ADC communication

After the production and testing of version 0.5 of the instrument design, one of the issues discovered was that the ADC used during development, was in fact not suitable for the task. Due to the nature of the analog sensor output and the amplification model developed (see section 6.3), the ADC conversions are performed as a differential measurement of the voltage, with $V_{REF125} = 1.25V$ as the negative input. This means that the ADC must have true differential inputs, able to measure both negative and positive differential voltages. As it turns out, the LTC1864L ADC does not support inputs with IN+ lower than IN-, and in its stead a pin-compatible component was employed, the Analog Devices AD7684. The communication with this device is also based on three signals (SPI/I^{2}C), so the only change needed was the actual state machine in VHDL controlling the communication with the ADC. The timing diagram for this communication is shown in figure 7.5 and the flow-chart of the state machine employed is shown in figure 7.6. The actual VHDL code used can be found in appendix C.

As the maximum allowed serial clock for the AD7684 is only 2.9 MHz, the system clock of 5.5Mhz could not be used directly. To enable clock compatibility with other blocks like *CRU_CONTROL* (see figure 7.4) which controls the *StartConv* signal, the 5.5Mhz system clock was used as the driving clock for the ADC state machine. The ADC clock was created by inverting the clock every rising edge of the system clock. As a result the pre-data state is actually a state for the sole purpose of waiting for the falling edge of the adc clock.

To keep options open for choosing between the LTC1864L and the AD7684 AD converters, a configurable solution was implemented, where zero-Ohm resistors select whether the IN- pin of the ADC should be connected to V_{REF125} or to GND (see Sensor Block in appendix D).

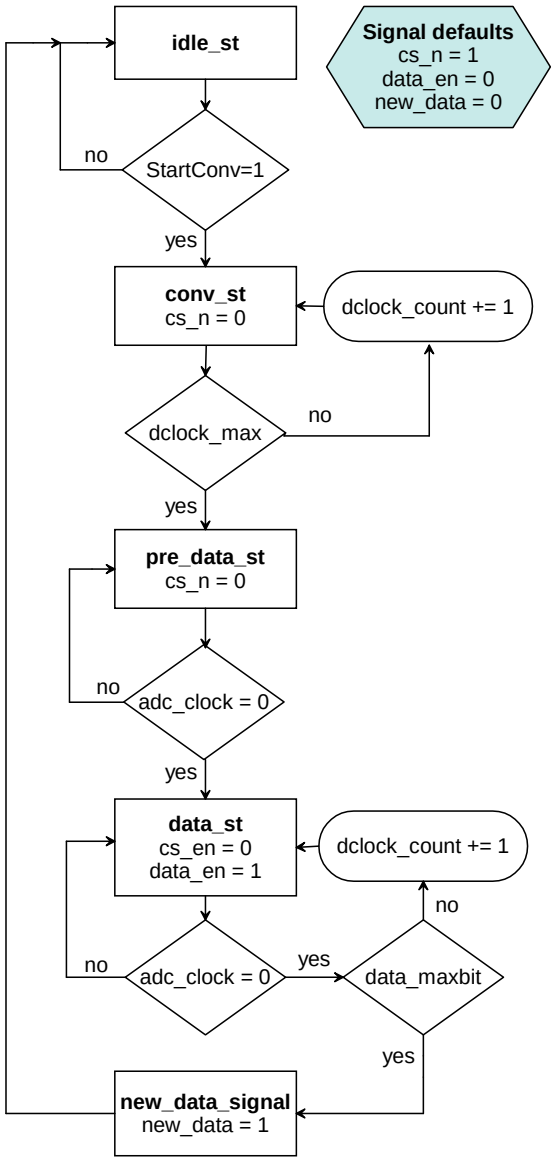

Figure 7.6: Flow chart illustrating the state machine used for serial communication with the AD7684.

7.6 Jumper configuration

Two jumpers connectors were included to serve as digital inputs to the CPLD. They were used for configuration of the instrument during testing, and also allowed for one code to be developed, while two different configurations for the communication were applied for the two cards prone for flight. (see section 5.1.2).

7.7 SR-Circuit

The SR sircuit was implemented mostly as in the development testing. The difference on the final design is that only 2.5V is available from the I/O port of the CPLD and therefore the first cMOS inverter stage was replaced by a fast opamp placed in a comparator configuration. The comparator is configured to switch to 3V when the input is higher than 1.25V and to 0V when lower.

7.8 Sensor and amplifier stages

The Amplifier stage was implemented in a similar way to how it was tested during development, the difference being that a higher gain was applied to take advantage of a larger part of the ADC dynamic range. A gain of $A_{TOT} = A1 * A2 = 27 * 23 = 621$ was used for the flight cards.

7.9 Lowpass Anti-aliasing filter

The lowpass filter was created using the application FilterPro from Texas Instruments. It is implemented in a Sallen Key Topology, with a passband of 0-100 Hz. This is part of the reason why a 2.9kHz sampling frequency was chosen, as Shannon's Theorem tells us that to avoid aliasing of a signal, we need to sample at a frequency twice the signal frequency. With the SallenKey filter implemented (4th order Chebyshev with allowed passband ripple of 0.004dB), we will be down to approximately -67dB at 1400Hz, which is less than half the sampling rate.

Chapter 8

Instrument Analysis

8.1 The setup

The instrument analysis takes place on a single-axis rate table (with tilt axis) of type Ideal Aerosmith 1291BR. As there is no external known magnetic field available as a precise reference, this is not a full calibration per se. The earth magnetic field is used as reference, and as such, the reference we have is only a model (IGRF). For the calibration, three sets of measurements are taken. For every measurement, one of the sensitive axes of the magnetometer is aligned with the axis around which the table rotates (see figure 8.1)

All axes on both cards were spun at different speeds both clockwise and counterclockwise, as well as in a sweep configuration (-2000 degrees per second to +2000 degrees per second). Unfortunately there was not enough time to analyse the data from the calibration. A vast dataset for both flight cards is available to compare to the modelled earth magnetic field for the date of calibration, but that could not be included in this thesis due to time limitations. Figures 8.2 and 8.3 are included as examples of how the calibration data can be used to evaluate the instrument performance and flight measurement data.

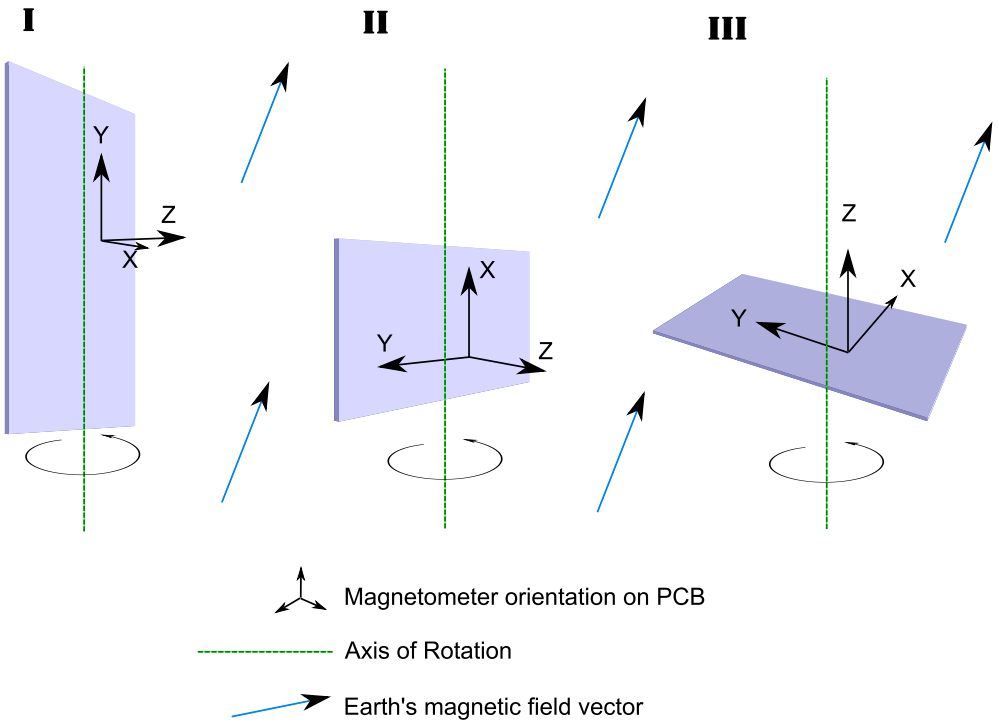

Figure 8.1: Magnetometer Calibration Setup.

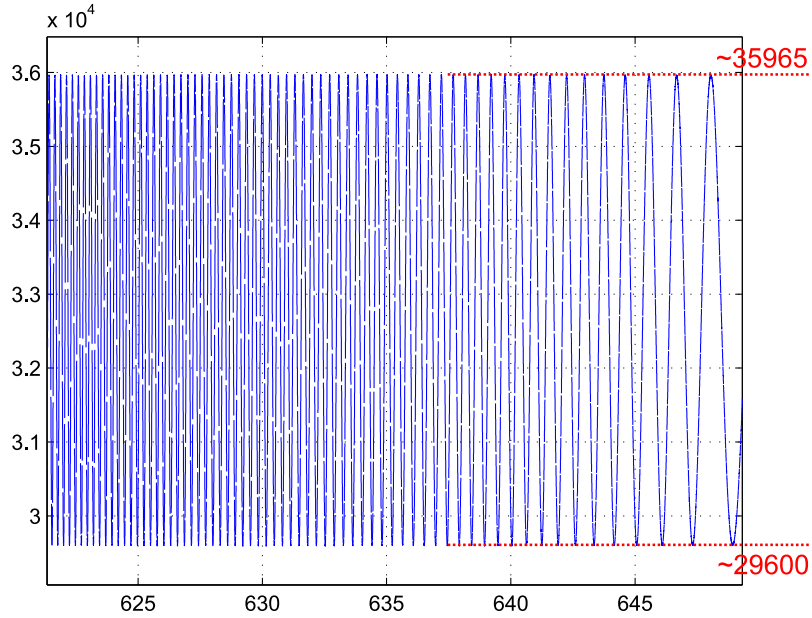

Figure 8.2: Plot of z-axis data from Card I doing a full sweep. Sensor is aligned in Mode II as shown on figure 8.1. Using the maximum and minimum values, we can find the middle bit $(35965 + 29600)/2 = 32782.5$, and by subtracting the ideal middle bit $2^{16}/2 = 32768$ we find the offset to be 32782,5-32768 = 14,5bits, which in this case is very small. The approximate measured intensity is given by the bit value $35965 - 32782.5 = 3182,5$. From the IGRF model values for Oslo on the calibration date, a horizontal intensity of $151.4 mgauss$ was expected. Using the figures for sensitivity from section 4.3 ($2.5\mu V/mgauss$), together with the amplification of 621, we expect a measured voltage of $151.4 mgauss * 2.5\mu V/mgauss * 621 = 234428\mu V$. The ADC resolution of $5V/2^{16} bits = 76,29 uV/bit$ means we expect a bit value of $234428\mu V/(76,29\mu V/bit) = 3079$.

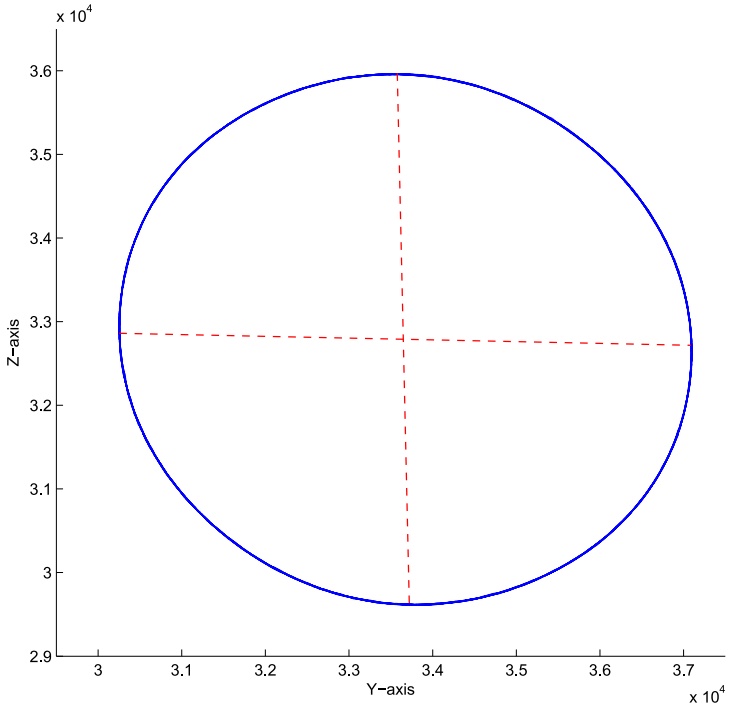

Figure 8.3: 3D plot of all three axes from Card I spinning clockwise at a rate of 1080 degrees/s. Sensor is aligned in Mode II as shown on figure 8.1. The plot is projected onto the Y-Z plane, meaning we only see the y- and z-axis components. As the rotations are made about the third axis (x), the vector values throughout time (around 37 full rotations) make a circle. The optimal result would be a perfect thin circle. We notice a slight skew from the perfect circle. We also notice a deviation from the zero-offset value of 32768 in the Y-axis. Further analysis must be carried out to investigate the reasons for these deviations. Data is available at a multitude of spins rates in both directions for all axes for both flight instruments, and can be used when interpreting measurements from the ICI-3 flight.

Chapter 9

Integration

Integration takes place at Andøya Rocket Range in the north of Norway. The first integration was a largely mechanical integration, where the instrument box was tested to fit in the payload structure, and the layout of pins and connectors were checked to make sure everything is in order. This was performed during September 2010. The second integration was held in August 2011, and this time, the interface and communication with the encoder was tested. The whole payload can be seen on figure 9.1, while figure 9.2 shows a photo of the instrument after successful integration, screwed tight with Loctite. The red marks show that they are fixed for good.

After the integration at Andøya, the Payload was brought to Narvik for environmental testing like vibration and temperature cycling. The last update from Narvik (environment tests are still going on as this thesis is submitted) was that the vibration testing was a success, and as expected from a solid state magnetic field sensor, no anomalies in the communication or measurements were observed.

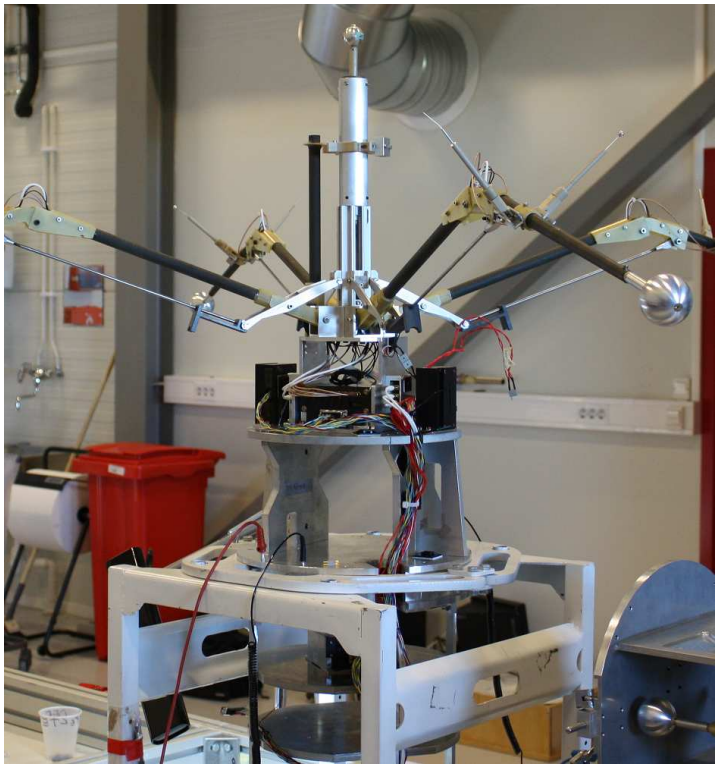

Figure 9.1: The nosecone of the rocket with the booms deployed. The SRADS MAG instrument sits in the nosecone structure of the rocket.

Figure 9.2: The instrument box fixed to the payload structure with Locktite, as indicated by the red marks on the screws. The grounding is done through pin6 on the DSUB-connector and forced in contact with the chassis through a screw. One jumper is glued in place on the topmost card in the box. This configures the card for communication with the TX2-slave encoder.

Chapter 10

Conclusions and Future work

Not everything could be tested during the limited amount of time available for this project. The flight has yet to happen at the time of writing and the measurement data taken at flight will be correlated with other instruments on board the ICI-3 (there are 3 different magnetometer instruments on board, including the housekeeping commercial magnetometer). The simplified calibration shows that the offset of the instruments is good, but not perfect. This is in large part due to imprecise values of the resistors used in amplification in the various stages, which give a lot of room for errors. The linearity of the measurements seems good to the naked eye, according to the 3D plot, but more analysis of the calibration data as well as the flight data must be performed before a conclusion can be reached.

10.1 Set-Reset solution

The Set-Reset circuit used in this thesis is dependent on a 3V power supply. As of the newest HMC1043 datasheet from Honeywell, the requirements for Reset current have been adjusted, and we might see an improvement in sensitivity and accuracy of measurements if the reset current is stronger. There are techniques to increase the voltage beyond 3V which could be applied in a 3V system, if it is found that the Reset Circuit needs more work. As mentioned in the SR development section, excessive temperature swings might also make the measurements sub-optimal; this is another motivation for further investigation.

10.2 Switching Feedback circuit

The Switching Feedback circuit is another part of this design that can be investigated deeper and there is probably good room for improvement. Using faster and more accurate operational amplifiers, or using different opamps at different stages, for example the OPAx376 at the first part of the dual-stage amplifier, and a faster, less precise opamp at the second stage, performance from the SWFB circuit could be improved. As such this work is to regard as a prototyping of this circuit principle, which yielded satisfiable results.

10.3 Data analysis

A large dataset from calibration of this instrument is available, and data from the flight will also provide a database of valuable data for the further analysis and development of this work.

List of abbreviations

ADC	Analog to Digital Converter. A device converting an analog voltage to a digital number.
AMR	Anisotropic Magneto-Resistive
ARR	Andøya Rocket Range
CMOS	Complementary Metal-Oxide-Semiconductor
CPLD	Complex Programmable Logic Device
ECEF	Earth-centered Earth Fixed. A reference coordinate system.
ECI	Earth-centered Inertial. A reference coordinate system.
FBGA	Fineline Ball Grid Array. A surcafe-mounted IC packaging standard
FET	Field-Effect Transistor
FOV	Field of View
GBP	Gain Bandwidth Product
GPS	Global Positioning System
I^2C	Inter-Integrated Circuit. A serial communication standard.
IGRF	International Geomagnetic Reference Field
LED	Light-emitting diode
MBGA	Micro Fineline Ball Grid Array
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MSOP-8	8-pin Mini Small Outline Package. A surcafe-mounted IC packaging standard
PDU	Power Distribution Unit
SFB	Spacecraft Body Frame. A reference coordinate system.

SOIC-8	8-pin Small Outline Integrated Circuit. A surcafe-mounted IC packaging standard
SPI	Serial Peripheral Interface. A serial communication standard.
SRADS	Sounding Rocket Attitude Determination System
TAM	Three-Axis Magnetometer
TEC	Total Electron Content
TQFP	Thin Quad Flat Pack. A surcafe-mounted IC packaging standard
UiO	University of Oslo

References

[1] Bekkeng, J.K. (2007). Prototype Development of a Low-Cost Sounding Rocket Attitude Determination System and an Electric Field Instrument. PhD thesis, University of Oslo.

[2] Bekkeng, Jan Kenneth. Prototyputvikling av E-felt eksperiment for små sonderaketter. Master Thesis. University of Oslo.

[3] Bekkeng, J.K., W. Boji, J. Moen (2005). Development of miniaturised low cost attitude determination system for sounding rockets. ESA Publications Division.

[4] Bekkeng, T.A., K. S. Jakobsen, J.K. Bekkeng, A. Pedersen, T. Lindem, J.-P. Lebreton, J. I. Moen. Design of a multi-needle Langmuir probe system

[5] Bekkeng, T.A. Prototype Development of a Multi-Needle Langmuir Probe System. Master Thesis. University of Oslo.

[6] Feng, J.S.Y., L.T.Romankiw and D.A.Thompson(1977). "Magnetic Self-Bias in the Barber Pole MR Structure". IEEE Transactions on Magnetics. Vol 13. Issue 5.

[7] Genish, Isaschar, Yevgeny Kats, Lior Klein, James W. Reiner, M. R. Beasley (2004). "Paramagnetic anisotropic magnetoresistance in thin films of SrRuO3". Journal of Applied Physics. Vol 95. 681-2004.

[8] Lanzerotti, Lous J. Space Weather Effects on Communications. Center for Solar-Terrestrial Research, New Jersey Institute of Technology.

[9] Lenz, James E.(1990). "A Review of Magnetic Sensors" Invited Paper. Proceedings of the IEEE. Vol 78. Issue 6. Current Version: 06 august 2002.

[10] Magnes, Werner, Marina Díaz-Michelena Future Directions for Magnetic Sensors for Space Applications

[11] Oreddson, Martin. Electrical Power System for the CubeSTAR Nanosatellite. Master Thesis. University of Oslo.

[12] Sollien, M.(2006). Prototyputvikling av digital solsensor for sonderaketter. Master Thesis. University of Oslo.

[13] Tresvig. Johan L, Tore Andre Bekkeng, Torfinn Lindem. CubeSTAR - A Nanosatellite for Space Weather Monitoring. University of Oslo.

[14] Wertz, James R., Wiley J. Larson (1999). Space Mission Analysis and Design. Third Edition.

[15] Altera. MAX II Device Handbook

[16] Analog Devices. AD7684 Datasheet.

[17] Andøya Rocket Range(2007). Hotel Payload - Instrument/PCM endoder interfacing recommendations and data handling. Revision C.

[18] Andøya Rocket Range (2008). Flight Requirements Plan, ICI-2 Campaign.

[19] Andøya Rocket Range(2009). PCM format budget for ICI-3. Separate documents for TX1 and TX2.

[20] Honeywell (2009). 3-Axis Magnetic Sensor HMC1043. Technical data sheet. Rev E.

[21] Honeywell. "Application of Magnetic Position Sensors". Application note AN211.

[22] Honeywell. "Handling Sensor Bridge Offset". Application Note AN212.

[23] Honeywell. "Set/Reset Function for Magnetic Sensors". Application note AN213.

[24] Honeywell. "Magnetic Sensors Product Catalog". Technical Article.

[25] Honeywell. "Magnetic Sensor Overview". Technical Article.

[26] International Rectifier. IRF7317 Datasheet. PD - 9.1568B.

[27] Hussein, Abid. Microchip Technology. Driving Power MOSFETs in High-Current, Switch Mode Regulators. Application note AN786.

[28] Linear Technology (2001). LTC1864 Datasheet.

[29] The CubeSat Program, California Polytechnic State University. "CubeSat Design Specification". Revision 12

[30] Georgia State University. "Hall Voltage for Positive Charge Carriers". HyperPhysics web resource. http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/hall.html

[31] NASA Images. http://www.nasaimages.com

[32] University of Melbourne. Fluxgate Magnetometer. Web resource. http://www.earthsci.unimelb.edu.au/ES304/MODULES/MAG/NOTES/fluxgate.html

[33] Wikimedia Commons. http://commons.wikimedia.org

Appendix A

ADC test code

A.1 Top file

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity adc_ctrl is
  port (
    -- global signals
    mclk    : in std_logic;   -- 66Mhz clock from devboard
    reset_n : in std_logic;   -- global reset-signal, if any

    -- ADC interface
    sck   : out std_logic;    -- serial clock
    conv  : out std_logic;    -- conversion control signal
    sdo   : in std_logic;     -- serial data

    -- LED output
    led_n : out std_logic_vector(1 to 4)
  );
end entity adc_ctrl;

architecture rtl of adc_ctrl is

  -- Clock divider, divides frequency by 20
  component clkdiv20 is
    port (
      mclk    : in std_logic;
      clk     : out std_logic
    );
  end component clkdiv20;

  -- Generic counter
  component countn is
    generic (
      msb       : in integer := 7;
      threshold : in unsigned
      );
    port (
      clk       : in  std_logic;
      rst       : in  std_logic;
      --count     : out std_logic_vector(n-1 downto 0);
      maxcount  : out std_logic
    );
  end component countn;

  -- 16-bit transparent shift register
```

```
  component shift16 is
    port(
      clk       : in std_logic;
      rst       : in std_logic;
      enable    : in std_logic;
      indata    : in std_logic;
      outdata   : out std_logic_vector(15 downto 0)
    );
  end component shift16;

  type adc_ctrl_states is (powerup_st0, powerup_st1, idle_st, conv_st,
      data_receive_st);
  signal adc_st             : adc_ctrl_states;

  signal new_data           : std_logic;
  signal adc_data           : std_logic_vector (15 downto 0);
--  signal adc_data_2bit      : std_logic_vector (1 downto 0);
  signal start_conv         : std_logic;
  signal sck_en             : std_logic;
  signal rst                : std_logic;
  signal clk                : std_logic;
  signal led_i              : std_logic_vector (1 to 4);

  constant bin1second       : unsigned (21 downto 0)  := "1100101100111111010000";
      -- ett sekund med 3,33MHz clk
  constant binhalfsecond    : unsigned                := "110010110011111101000";
      -- et halvt sekund med 3,33MHz clk
  constant binquartersecond : unsigned                := "11001011001111110100";  --
      et halvt sekund med 3,33MHz clk
  constant bin62500us       : unsigned (17 downto 0)  := "11001011001111101";
      -- 62,5 ms
  constant bin30030ns       : unsigned (6 downto 0)   := "1100100";
      -- 33,3 kHz = 30,030 us
  constant bin15015ns       : unsigned                := "110010";
      -- 66,6 kHz = 15,015 us

begin

  -- This timer outputs 'sample_timer_done' signal, which initiates sampling
  -- Various timing examples are provided commented
  SAMPLE_TIMER: countn
  generic map (
    -- msb       => 21,
    -- threshold => bin1second
    -- msb       => 20,
    -- threshold => binhalfsecond
    -- msb       => 19,
    -- threshold => binquartersecond
    -- msb       => 17,
    -- threshold => bin62500us
    -- msb       => 6,
    -- threshold => bin30030ns
    msb       => 5,
    threshold => bin15015ns
  )
  port map (
    clk       => clk,
    rst       => rst,
    maxcount  => start_conv
  );

  -- Divides the 66Mhz clock by 20 (3,3Mhz) to resemble the 3,33Mhz clock of ICI-3
  CLKDIV20_1: clkdiv20
  port map(
    mclk => mclk,
    clk  => clk
  );
```

```
107   -- LED DISPLAY
      LED_REGISTER: process(rst,clk,adc_data,new_data)
      begin
        if rst = '1' then
111       led_i <= (others => '0');
        elsif rising_edge(clk) and new_data = '1' then
          led_i <= adc_data (15 downto 12);
        end if;
115   end process LED_REGISTER;
      led_n (1 to 4) <= not led_i (1 to 4);

      SHIFT16_1 : shift16
119   port map (
        clk     => clk,
        rst     => rst,
        enable  => sck_en,
123     indata  => sdo,
        outdata => adc_data
      );
      sck <= clk when sck_en = '1' else 'Z';
127
      -- Controls the states of the ADC interface
      -- COMB controls outputs from states
      -- SEQ controls the sequential logic
131   ADC_STATE_COMB: process(adc_st)
      begin
        case adc_st is
          when powerup_st0 =>   -- The two powerup states output rst signal high
135         rst <= '1';
            conv <= '0';
            sck_en <= '0';
          when powerup_st1 =>
139         rst <= '1';
            conv <= '0';
            sck_en <= '0';
          when idle_st =>       -- This is the default state between measurements
143         rst <= '0';
            conv <= '0';
            sck_en <= '0';
          when conv_st =>       -- Initiate conversion and keep signal high until
              conversion done
147         rst <= '0';
            conv <= '1';
            sck_en <= '0';
          when data_receive_st => -- Enable serial clock to receive data
151         rst <= '0';
            conv <= '0';
            sck_en <= '1';
          when others =>            -- in case of undefined state all outputs low
155         rst <= '0';
            conv <= '0';
            sck_en <= '0';
        end case;
159   end process ADC_STATE_COMB;
      ADC_STATE_SEQ: process(clk, reset_n)
      variable bits_received : unsigned (3 downto 0);
      variable t_conv_count  : unsigned (3 downto 0);
163   begin
        if reset_n = '0' then
          adc_st <= powerup_st0;
        elsif rising_edge(clk) then
167       new_data <= '0';
          case adc_st is
            when powerup_st0 =>       -- The device should power-up into this state
              adc_st <= powerup_st1;
171         when powerup_st1 =>       -- Second powerup-state to ensure a minimum of one
                clock period of power_up time
              adc_st <= idle_st;
```

```
        when idle_st =>           -- This state waits for start_conv signal, then
            goes to conv_st
          t_conv_count := "0000";
          bits_received := "0000";
          if start_conv = '1' then
            adc_st <= conv_st;
          end if;
        when conv_st =>           -- Starts conversion and waits for conversion time
            t_conv, goes to data_receive
          if t_conv_count = "1111" then
            adc_st <= data_receive_st;
          end if;
          t_conv_count := t_conv_count + 1;
        when data_receive_st =>   -- Waits for 16 clock cycles to receive 16 bits
          if bits_received = "1111" then
            adc_st <= idle_st;
            new_data <= '1';
          end if;
          bits_received := bits_received + 1;
        when others =>
          adc_st <= idle_st;
      end case;
    end if;
  end process ADC_STATE_SEQ;

end architecture rtl;
```

codeADC/adc_ctrl.vhd

A.2 Clock Divider

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity clkdiv20 is
  port (
    mclk    : in std_logic;
    clk     : out std_logic
  );
end entity clkdiv20;

architecture rtl of clkdiv20 is

begin

  CLK_DIVIDER: process (mclk)
  variable count_i  : unsigned(3 downto 0) := (others => '0');
  variable clk_i    : std_logic := '0';
  begin
    -- if rst = '1' then
      -- clk <= '0';
      -- count_i := (others => '0');
    if rising_edge(mclk) then
      count_i := count_i + 1;
      if count_i = "1010" then  -- Teller til 10 oppadgående flanker
        clk_i := not clk_i;
        count_i := (others => '0');
      end if;
    end if;
    clk <= clk_i;
  end process CLK_DIVIDER;

end architecture rtl;
```

codeADC/clkdiv20.vhd

A.3 Generic Counter

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity countn is
  generic (
    msb       : in integer := 7; -- Antall bit i telleren, default 8
    threshold : in unsigned      -- Spesifisert tellelengde.
    );
  port (
    clk       : in  std_logic;
    rst       : in  std_logic;
--    count     : out std_logic_vector(n-1 downto 0);
    maxcount  : out std_logic
  );
end countn;

architecture rtl of countn is

signal count_i : unsigned(msb downto 0);

begin

  COUNTER:
  process (rst, clk, count_i)
  begin
    if(rst = '1') then
      count_i <= (others => '0');
      maxcount <= '0';
    elsif rising_edge(clk) then
      count_i <= count_i + 1;
      maxcount <= '0';
      if count_i = threshold then
        count_i <= (others => '0');
        maxcount <= '1';
      end if;
    end if;
  --count <= std_logic_vector(count_i);
  end process counter;

end architecture rtl;
```

codeADC/countn.vhd

A.4 16-bit Shift Register

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift16 is
  port (
    clk     : in std_logic;
    rst     : in std_logic;
    enable  : in std_logic;
    indata  : in std_logic;
    outdata : out std_logic_vector(15 downto 0)
  );
end entity shift16;

architecture rtl of shift16 is
  signal outdata_i : std_logic_vector(15 downto 0);

  begin

  outdata <= outdata_i;

  SHIFT_REG: process(clk, rst, enable)
  begin
    if rst = '1' then
      outdata_i <= (others => '0');
    elsif rising_edge(clk) and enable = '1' then
      outdata_i(0) <= indata;
      for i in 1 to 15 loop
        outdata_i(i) <= outdata_i(i-1);
      end loop;
    end if;
  end process SHIFT_REG;
end architecture rtl;
```

codeADC/shift16.vhd

Appendix B

Sensor test code

B.1 Top file

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mag_sr_test is
  port (
    -- global signals clock & reset
    mclk     : in std_logic;  -- master clock, 66Mhz from devboard

    -- outputs
    vsr      : out std_logic;           -- controls reset voltage Vsr
    demodulate_pos : out std_logic;     -- controls the demodulation switch, positive
    demodulate_neg : out std_logic;     -- controls the demodulation switch, negative
    led_n    : out std_logic_vector(1 to 4); -- controls LEDs on devboard
    clk_out : out std_logic             -- output for analysis
  );
end entity mag_sr_test;

architecture rtl of mag_sr_test is

  -- Clock divider, divides mclk frequency by 20
  component cru is
    port (
      mclk     : in std_logic;   -- external clock
      rst      : out std_logic;  -- internal reset
      clk      : out std_logic   -- internal clock, 3,3Mhz
    );
  end component cru;

  -- Generic counter
  component countn is
    generic (
      msb       : in integer := 7;
      threshold : in unsigned
      );
    port (
      clk       : in  std_logic;
      rst       : in  std_logic;
      maxcount  : out std_logic
    );
  end component countn;

  -- This component controls the voltage output that the Set-reset curcuit
  -- needs to perform the set-reset operation on the magnetometer HMC1043
  -- vsr stays high when idle, is dropped to low to execute SET, then pulled
```

```
  -- high again after 'spacing' clock cycles to perform the RESET function.
47  component mag_sr_logic is
      generic (
        -- These values control the timing of the set-reset-cycle and
        -- must be set up according to reset curcuit hardware (capacitor values)
51      spacing : in unsigned;  -- Number of clock periods (-1) separating set&reset
        spacing_count_msb : in integer := 7 -- MSB of spacing vector
      );
      port (
55      -- inputs
        rst : in std_logic;       -- asyncronous reset
        clk : in std_logic;
        start_sr : in std_logic; -- held high for one clock period to start an sr-cycle
59
        -- outputs
        vsr : out std_logic -- controls reset voltage Vsr
      );
63  end component mag_sr_logic;

    -- global internal rst & clock signals
    signal rst       : std_logic;
67  signal clk       : std_logic;

    -- mag_sr-specific signals
    signal start_sr : std_logic;
71
    -- led-specific signals
    signal switch_led : std_logic;
    signal led_i      : std_logic_vector (1 to 4);
75  signal led_mux    : std_logic_vector (1 downto 0);

    signal vsr_i : std_logic;

79  -- Timing constants
    -- 3,3Mhz/16384 = 201 Hz, T=4,98ms
    constant bin5ms     : unsigned (13 downto 0) := "11111111111111";
    -- 3,3Mhz/256 = 12.89kHz, T=155,15us
83  constant bin77us6   : unsigned (7 downto 0)  := "11111111";
    -- 3,3Mhz/512 = 6.4kHz, T=155,15us
    constant bin155us   : unsigned (8 downto 0)  := "111111111";
    -- 3,3Mhz/1024 = 3222,65625Hz, T=310,30us
87  constant bin310us   : unsigned (9 downto 0)  := "1111111111";
    -- 3,3Mhz/2048 = 1.6kHz, T=620,60us
    constant bin620us   : unsigned (10 downto 0) := "11111111111";
    -- 3,3Mhz/4096 = 800Hz, T= 1240us
91  constant bin1240us  : unsigned (11 downto 0) := "111111111111";
    -- 3,3Mhz/8192 = 400Hz, T= 2480us
    constant bin2480us  : unsigned (12 downto 0) := "1111111111111";
    -- 3,3Mhz/64 = 515,625kHz, T=19,39394us
95  constant bin19us    : unsigned (5 downto 0)  := "111111";
    -- ett sekund
    constant bin1second : unsigned (21 downto 0) := "1100100101101010011111";

99
  begin

    -- For analysis
103 vsr <= vsr_i;
    demodulate_pos <= not vsr_i;
    demodulate_neg <= vsr_i;

107 FILTER_CLK: process (clk,rst)
    variable filter_clk_count : unsigned (8 downto 0);
    begin
    if rst = '1' then
111     filter_clk_count := (others => '0');
      elsif rising_edge (clk) then
        filter_clk_count := filter_clk_count + 1;
```

```
    end if;
    clk_out <= std_logic(filter_clk_count(8));
  end process FILTER_CLK;

  -- Divides the 66Mhz clock by 20 (3,3Mhz) to approximate
  -- the 3,33Mhz clock of ICI-3 (T = 303,030303030303 ns)
  CRU1: cru
  port map(
    mclk => mclk,
    rst  => rst,
    clk  => clk
  );

  -- This timer controls the 'start_sr' signal,
  -- which is used to initiate a set-reset-cycle
  RESET_TIMER: countn
  generic map (
    -- msb       => 5,
    -- threshold => bin19us -- 515kHz sr-cycle rate
    -- msb       => 12,
    -- threshold => bin2480us -- 400Hz sr-cycle rate, 1024 cycles
    -- msb       => 11,
    -- threshold => bin1240us -- 800Hz sr-cycle rate, 1024 cycles
    -- msb       => 10,
    -- threshold => bin620us -- 1.6kHz sr-cycle rate
    msb       => 7,
    threshold => bin77us6 -- 12.89kHz sr-cycle rate
    -- msb       => 8,
    -- threshold => bin155us -- 6.4kHz sr-cycle rate
    -- msb       => 9,
    -- threshold => bin310us -- 3.2kHz sr-cycle rate
    -- msb       => 13,
    -- threshold => bin5ms -- 200Hz sr-cycle rate
    -- msb       => 21,
    -- threshold => bin1second -- 1Hz sr-cycle rate
  )
  port map (
    clk       => clk,
    rst       => rst,
    maxcount  => start_sr
  );

  -- This timer counts one second to cycle the leds every second.
  LED_TIMER: countn
  generic map (
    msb       => 21,
    threshold => bin1second
    )
  port map (
    clk       => clk,
    rst       => rst,
    maxcount  => switch_led
  );

  MAG_SR_LOGIC_0: mag_sr_logic
  generic map (
    -- Spacing, current clock is 3.3Mhz => T = 303ns (300ns for ICI-3)
    -- spacing           => "1100011111111", -- (+1) = 6400 clock cycles = 1.94 ms
    -- spacing_count_msb => 12
    -- spacing           => "11000111111", -- (+1) = 1600 clock cycles = 485 us
    -- spacing_count_msb => 10
    -- spacing           => "1100011111", -- (+1) = 800 clock cycles = 242,4 us
    -- spacing_count_msb => 9
    -- spacing           => "111111111", -- (+1) = 512 clock cycles
    -- spacing_count_msb => 8
    -- spacing           => "11111111", -- (+1) = 256 clock cycles
    -- spacing_count_msb => 7
    spacing           => "1111111", -- (+1) = 128 clock cycles
```

```
    spacing_count_msb => 6
    -- spacing           => "1111111111", -- (+1) = 1024 clock cycles
    -- spacing_count_msb => 9
    -- spacing           => "11111111111", -- (+1) = 2048 clock cycles
    -- spacing_count_msb => 10
    -- spacing           => "111111111111", -- (+1) = 4097 clock cycles
    -- spacing_count_msb => 11
    -- spacing           => "11000111", -- (+1) = 200 clock cycles = 60,6 us
    -- spacing_count_msb => 7
    -- spacing           => "1100011", -- (+1) = 100 clock cycles = 30,3 us
    -- spacing_count_msb => 6
    -- spacing           => "10011", -- (+1) = 20 clock cycles = 6,06 us
    -- spacing_count_msb => 4
    -- spacing           => "11101", -- (+1) = 30 clock cycles = 9,09 us
    -- spacing_count_msb => 4
  )
  port map (
    rst       => rst,
    clk       => clk,
    start_sr  => start_sr,
    vsr       => vsr_i
  );


  ------------------------------------------------------
  -- For output to leds --------------------------------
  ------------------------------------------------------
  LED_TEST: process(clk, rst, switch_led)
  variable test_count : unsigned (1 downto 0) := (others => '0');
  begin
    if rst = '1' then
      test_count := (others => '0');
    elsif falling_edge(clk) and switch_led = '1' then
      test_count := test_count + 1;
    end if;
    led_mux <= not std_logic_vector(test_count);
  end process LED_TEST;
  with led_mux select
      led_i <= "0001" when "00",
               "0010" when "01",
               "0100" when "10",
               "1000" when "11",
               "0000" when others;
  led_n <= not led_i;
  ------------------------------------------------------
  ------------------------------------------------------

end architecture rtl;
```

codeSR/mag_sr_test.vhd

B.2 Clock Reset Unit

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity cru is
  port (
    mclk    : in std_logic;
    rst     : out std_logic;
    clk     : out std_logic
    );
end entity cru;

architecture rtl1 of cru is

  signal rst_i : std_logic;

  component rst_unit is
    port(
      mclk : in std_logic;
      rst : out std_logic
    );
  end component rst_unit;

  -- component clkdivn is
    -- port
      -- (
      -- clk_in  : in  std_logic;
      -- reset   : in  std_logic;
      -- clk_out : out std_logic
      -- );
  -- end component clkdivn;



  component clkdiv20 is
    port (
      mclk    : in std_logic;
      rst     : in std_logic;
      clk     : out std_logic
    );
  end component clkdiv20;

  begin

  rst <= rst_i;

  RST_UNIT_1: rst_unit
  port map(
    mclk => mclk,
    rst => rst_i
  );

  -- CLKDIVN_1: clkdivn
  -- port map(
    -- clk_in  => mclk,
    -- reset   => rst_i,
    -- clk_out => clk
  -- );

  CLKDIV20_1: clkdiv20
  port map(
    mclk => mclk,
    rst => rst_i,
    clk => clk
  );
```

```
67 end architecture rtl1;
```

codeSR/cru.vhd

B.3 Reset Unit

```
1  library ieee;
   use ieee.std_logic_1164.all;
   use ieee.numeric_std.all;

5  entity rst_unit is
     port(
       mclk : in std_logic;
       rst : out std_logic
9    );
   end entity rst_unit;

   architecture rtl of rst_unit is
13
     -- powerup state machine types and signals
     type powerup_states is (powerup_st0, powerup_st1, idle_st);
     signal powerup_st : powerup_states;
17
     begin

     -- Controls the powerup phase of the CPLD and internal reset signal
21   -- COMB controls outputs from states
     -- SEQ controls the sequential logic
     POWERUP_STATE_COMB: process(powerup_st)
     begin
25     case powerup_st is
       -- The two powerup states output rst signal high
         when powerup_st0 =>
           rst <= '1';
29       when powerup_st1 =>
           rst <= '1';
       -- This is the default state between measurements
         when idle_st =>
33         rst <= '0';
         when others =>
           rst <= '1';
       end case;
37   end process POWERUP_STATE_COMB;
     POWERUP_STATE_SEQ: process(mclk)
     begin
       if rising_edge(mclk) then
41       case powerup_st is
-- According to the CPLD datasheet, the device will powerup into this state
           when powerup_st0 =>
             powerup_st <= powerup_st1;
45-- A second powerup-state ensures a minimum of one clock period of power_up time
           when powerup_st1 =>
             powerup_st <= idle_st;
-- After powerup, the device will enter this state and stay here
49         when idle_st =>
             powerup_st <= idle_st;
-- In case of undefined state, the device goes into one powerup-state
           when others =>
53           powerup_st <= powerup_st1;
         end case;
       end if;
     end process POWERUP_STATE_SEQ;
57
   end architecture rtl;
```

codeSR/rst_unit.vhd

B.4 Clock Divider

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity clkdiv20 is
  port (
    rst     : in std_logic;
    mclk    : in std_logic;
    clk     : out std_logic
    );
end entity clkdiv20;

-- architecture rtl1 of clkdiv20 is

  -- begin

  -- CLK_DIVIDER: process (rst, mclk)
  -- variable count_i  : unsigned(4 downto 0) := (others => '0');
  -- begin
    -- if rst = '1' then
      -- count_i := (others => '0');
      -- clk <= '0';
    -- elsif rising_edge(mclk) then
      -- count_i := count_i + 1;
      -- if count_i = "01010" then
        -- clk <= '1';
      -- elsif count_i = "10100" then
        -- clk <= '0';
        -- count_i := (others => '0');
      -- end if;
    -- end if;
  -- end process CLK_DIVIDER;
-- end architecture rtl1;

architecture rtl2 of clkdiv20 is

  begin

  CLK_DIVIDER: process (rst, mclk)
  variable count_i  : unsigned(3 downto 0) := (others => '0');
  variable clk_i    : std_logic := '0';
  begin
    if rst = '1' then
      count_i := (others => '0');
      clk_i := '0';
    elsif rising_edge(mclk) then
      count_i := count_i + 1;
      if count_i = "1010" then
        clk_i := not clk_i;
        count_i := (others => '0');
      end if;
      clk <= clk_i;
    end if;
  end process CLK_DIVIDER;

end architecture rtl2;
```

codeSR/clkdiv20.vhd

B.5 MAG SR Logic

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-- This component controls the voltage output that the Set-reset curcuit
-- needs to perform the set-reset operation on the magnetometer HMC1043
-- vsr stays high when idle, is dropped to low to execute SET, then pulled
-- high again after "spacing" clock cycles to perform the RESET function.
entity mag_sr_logic is
  generic (
    -- These values control the timing of the set-reset-cycle and
    -- must be set up according to reset curcuit hardware (capacitor values)
    spacing : in unsigned; -- Number of clock periods (-1) separating set&reset
    spacing_count_msb : in integer := 7 -- MSB of spacing vector
  );
  port (
    -- inputs
    rst : in std_logic;      -- asyncronous reset
    clk : in std_logic;
    start_sr : in std_logic; -- held high for one clock period to start an sr-cycle

    -- outputs
    vsr : out std_logic -- controls reset voltage Vsr
  );
end entity mag_sr_logic;

architecture rtl of mag_sr_logic is

  type vsr_states is (vlow, vhigh);
  signal vsr_st : vsr_states;

  begin

  -- Active low reset signal
  -- SR_STATE_MACHINE: process (rst, clk, start_sr)
  -- variable count_i : unsigned (spacing_count_msb downto 0);
  -- begin
    -- if rst = '1' then
      -- count_i := (others => '0');
      -- vsr_st <= vhigh;
    -- elsif falling_edge(clk) then
      -- case vsr_st is
        -- when vhigh =>
          -- count_i := (others => '0');
          -- if start_sr = '1' then
            -- vsr_st <= vlow;
          -- end if;
        -- when vlow =>
          -- This check will find that the counter is 0 after one clock period
          -- because one clock cycle passes before this state is entered
          -- if count_i = spacing then
            -- vsr_st <= vhigh;
          -- else
            -- count_i := count_i + 1;
          -- end if;
        -- when others =>
          -- vsr_st <= vhigh;
      -- end case;
    -- end if;
  -- end process SR_STATE_MACHINE;
  -- with vsr_st select
    -- vsr <= '1' when vhigh,
           -- '0' when others;

  -- Active high reset signal
```

```
  SR_STATE_MACHINE: process (rst, clk, start_sr)
  variable count_i : unsigned (spacing_count_msb downto 0);
68 begin
    if rst = '1' then
      count_i := (others => '0');
      vsr_st <= vlow;
72  elsif falling_edge(clk) then
      case vsr_st is
        when vlow =>
          count_i := (others => '0');
76        if start_sr = '1' then
            vsr_st <= vhigh;
          end if;
        when vhigh =>
80      -- This check will find that the counter is 0 after one clock period
        -- because one clock cycle passes before this state is entered
        if count_i = spacing then
            vsr_st <= vlow;
84        else
            count_i := count_i + 1;
          end if;
        when others =>
88        vsr_st <= vlow;
      end case;
    end if;
  end process SR_STATE_MACHINE;
92 with vsr_st select
    vsr <= '1' when vhigh,
           '0' when others;

96 end architecture rtl;
```

codeSR/mag_sr_logic.vhd

B.6 Generic Counter

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity countn is
  generic (
    msb       : in integer := 7; -- MSB of counter
    threshold : in unsigned      -- Counting length
    );
  port (
    clk       : in  std_logic;
    rst       : in  std_logic;
    maxcount  : out std_logic
  );
end countn;

architecture rtl of countn is

begin

  COUNTER:
  process (rst, clk)
    variable count_i : unsigned(msb downto 0);
  begin
    if rst = '1' then
      count_i := (others => '0');
      maxcount <= '0';
    elsif rising_edge(clk) then
      if count_i = threshold then
        count_i := (others => '0');
        maxcount <= '1';
      else
        count_i := count_i + 1;
        maxcount <= '0';
      end if;
    end if;
  end process counter;

end architecture rtl;
```

codeSR/countn.vhd

Appendix C

Final VHDL code

C.1 Top file

```
-- This is the top file for the MAXIIG CPLD
-- on the SRADS_MAG_v50 card
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity SRADS_MAG is
  port
  (
    -- PCM Endoder I/O --
    SCLK_N          :   in std_logic;   -- P12/GCLK0
    MINF_GLOB_N     :   in std_logic;   -- P14/GCLK1
    MINF_N          :   in std_logic;   -- P15
    MAJF_GLOB_N     :   in std_logic;   -- P62/GCLK2
    MAJF_N          :   in std_logic;   -- P61
    GATE_GLOB_N     :   in std_logic;   -- P64/GCLK3
    GATE_N          :   in std_logic;   -- P66
    DATA            :   out std_logic;  -- P7

    -- AUX I/O --
    JP1             :   in std_logic;   -- P20
    JP2             :   in std_logic;   -- P21
    AUX_IO_1        :   out std_logic;  -- P52
    AUX_IO_2        :   out std_logic;  -- P51
    AUX_IO_3        :   out std_logic;  -- P53
    AUX_IO_4        :   out std_logic;  -- P54
    AUX_DATA_OUT    :   out std_logic;  -- P26

    -- SET/RESET control --
    Vsr             :   buffer std_logic;  -- P28
    polarity_demod  :   buffer std_logic;  -- P47

    -- ADC control I/O --
    X_SDO           :   in std_logic;   -- P40
    Y_SDO           :   in std_logic;   -- P30
    Z_SDO           :   in std_logic;   -- P35
    X_SCK           :   out std_logic;  -- P38
    Y_SCK           :   out std_logic;  -- P29
    Z_SCK           :   out std_logic;  -- P34
    X_CONV          :   out std_logic;  -- P41
    Y_CONV          :   out std_logic;  -- P33
    Z_CONV          :   out std_logic   -- P36
  );
end entity SRADS_MAG;

```

```
architecture top of SRADS_MAG is
  component ADC_interface is
    port (
      -- ADC control input
      a_rst      : in std_logic;
      s_rst      : in std_logic;
      mclk       : in std_logic;
      StartConv : in std_logic;

      -- ADC control
      CONV   : out std_logic;
      SCK    : out std_logic;

      -- Shiftreg control
      X_Shift_en   : out std_logic;
      Y_Shift_en   : out std_logic;
      Z_Shift_en   : out std_logic;

      -- UART control
      new_data     : out std_logic
    );
  end component ADC_interface;

  component ADC_interface2 is
    port
    (
      -- ADC control input
      a_rst      : in std_logic;
      s_rst      : in std_logic;
      mclk       : in std_logic;
      StartConv : in std_logic;

      -- ADC control
      CS_N       : out std_logic;
      DCLOCK     : buffer std_logic;

      -- Shiftreg control
      X_Shift_en   : out std_logic;
      Y_Shift_en   : out std_logic;
      Z_Shift_en   : out std_logic;

      -- UART control
      new_data     : out std_logic
    );
  end component ADC_interface2;

  component shift16 is
    port
    (
      -- Control inputs
      a_rst    : in std_logic;
      mclk     : in std_logic;
      enable   : in std_logic;

      -- 1-bit data input
      indata   : in std_logic;

      -- 16-bit data output
      outdata : out std_logic_vector(15 downto 0)
    );
  end component shift16;

  component PCM_interface is
    port
    (
      -- Control inputs
      PCM_enable     : in std_logic;
      a_rst          : in std_logic;
```

```
      s_rst          : in std_logic;
      mclk           : in std_logic;
      SCLK_N         : in std_logic;
      GATE           : in std_logic;
      current_axis   : in integer range 0 to 2;

      -- Data to send
      OutData_X      : in std_logic_vector(15 downto 0);
      OutData_Y      : in std_logic_vector(15 downto 0);
      OutData_Z      : in std_logic_vector(15 downto 0);

      -- Data output
      DATA           : out std_logic
    );
  end component PCM_interface;

  component UART_Transmitter is
    port
    (
      -- Control inputs
      UART_enable    : in std_logic;
      a_rst          : in std_logic;
      s_rst          : in std_logic;
      mclk           : in std_logic;
      new_data       : in std_logic;
      --current_axis  : in integer range 0 to 2;

      -- Control data
      Outdata_X      : in std_logic_vector(15 downto 0);
      Outdata_Y      : in std_logic_vector(15 downto 0);
      Outdata_Z      : in std_logic_vector(15 downto 0);

      -- Data output
      AUX_DATA_OUT : out std_logic
    );
  end component UART_Transmitter;

  component cru_control
    port
    (
      -- External signals
      SCLK_N       : in std_logic;
      MINF       : in std_logic;
      MAJF       : in std_logic;
      GATE       : in std_logic;
      JP1        : in std_logic;
      JP2        : in std_logic;

      -- Clock and reset signals
      a_rst      : buffer std_logic;
      s_rst      : buffer std_logic;
      mclk       : buffer std_logic;

      -- Communication signals
      UART_enable      : buffer std_logic;
      PCM_enable       : buffer std_logic;
      current_axis     : out integer range 0 to 2;

      -- Magnetometer and ADC control
      Vsr              : out std_logic;
      polarity_demod   : out std_logic;
      StartConv        : out std_logic;

      Test_vector      : out std_logic_vector(3 downto 0)
    );
  end component cru_control;

  -- signal MINF_GLOB     : std_logic;
```

```
    signal MINF          : std_logic;
    -- signal MAJF_GLOB     : std_logic;
    signal MAJF          : std_logic;
    -- signal GATE_GLOB     : std_logic;
    signal GATE          : std_logic;

    -- Global control signals
    signal a_rst         : std_logic;
    signal s_rst         : std_logic;
    signal mclk          : std_logic;

    -- Interface signals
    signal UART_enable   : std_logic;
    signal PCM_enable    : std_logic;
    signal current_axis  : integer range 0 to 2;

    -- ADC-signals
    signal StartConv     : std_logic;
    signal new_data      : std_logic;
    signal DCLOCK        : std_logic;
    signal CS_N          : std_logic;
    -- signal SCK           : std_logic;
    -- signal CONV          : std_logic;

    -- Shift register signals
    signal X_Shift_en    : std_logic;
    signal Y_Shift_en    : std_logic;
    signal Z_Shift_en    : std_logic;

    signal X_Data_raw    : std_logic_vector(15 downto 0);
    signal Y_Data_raw    : std_logic_vector(15 downto 0);
    signal Z_Data_raw    : std_logic_vector(15 downto 0);

    signal X_Data        : std_logic_vector(15 downto 0);
    signal Y_Data        : std_logic_vector(15 downto 0);
    signal Z_Data        : std_logic_vector(15 downto 0);

    signal Test_vector   : std_logic_vector(3 downto 0);

begin

    -- For AD7684 _NON_ 2's complement (for realtime decoder view)
    X_Data(14 downto 0) <= X_data_raw(14 downto 0);
    Y_Data(14 downto 0) <= Y_data_raw(14 downto 0);
    Z_Data(14 downto 0) <= Z_data_raw(14 downto 0);
    X_Data(15) <= not X_data_raw(15);
    Y_Data(15) <= not Y_data_raw(15);
    Z_Data(15) <= not Z_data_raw(15);

    -- For AD7684 2's complement or LTC1864
    -- X_Data <= X_data_raw;
    -- Y_Data <= Y_data_raw;
    -- Z_Data <= Z_data_raw;

    -- Use normal I/O pin connected control signals
    MINF        <= not MINF_N;
    MAJF        <= not MAJF_N;
    GATE        <= not GATE_N;
    -- Global clock net connected signals may be tried if timing is a problem
    -- MINF     <= not MINF_GLOB_N;
    -- MAJF     <= not MAJF_GLOB_N;
    -- GATE     <= not GATE_GLOB_N;


    -- ***************************************************************************
    -- This process may be used for debugging the current_axis signal
    -- It uses two AUX I/O pins to display the 2bit value of current_axis
    -- ***************************************************************************
```

```
-- TEST_VECTOR0: process (mclk, current_axis)
-- begin
  -- if rising_edge(mclk) then
    -- case current_axis is
      -- when 0 =>
        -- AUX_IO_3 <= '0';
        -- AUX_IO_4 <= '0';
      -- when 1 =>
        -- AUX_IO_3 <= '0';
        -- AUX_IO_4 <= '1';
      -- when 2 =>
        -- AUX_IO_3 <= '1';
        -- AUX_IO_4 <= '0';
      -- when others =>
        -- AUX_IO_3 <= '1';
        -- AUX_IO_4 <= '1';
      -- end case;
  -- end if;
-- end process TEST_VECTOR0;

-- Set up which signals drive the AUX I/O pins.
AUX_IO_1  <=  MINF;
AUX_IO_2  <=  GATE;
AUX_IO_3  <=  StartConv;
AUX_IO_4  <=  Test_vector(0); -- Vsr

-- ADC_interface2 (AD7684)
-- All communication happens synchronously for all axes
X_SCK     <= DCLOCK;
Y_SCK     <= DCLOCK;
Z_SCK     <= DCLOCK;
X_CONV    <= CS_N;
Y_CONV    <= CS_N;
Z_CONV    <= CS_N;

-- ADC_interface (LT1864L)
-- All communication happens synchronously for all axes
-- X_SCK      <= SCK;
-- Y_SCK      <= SCK;
-- Z_SCK      <= SCK;
-- X_CONV     <= CONV;
-- Y_CONV     <= CONV;
-- Z_CONV     <= CONV;

-- Switch out applicable lines for switching ADC
-- ADC_INTERFACE_UNIT: ADC_interface
ADC_INTERFACE_UNIT: ADC_interface2
port map
(
  a_rst         => a_rst,
  s_rst         => s_rst,
  mclk          => mclk,
  StartConv     => StartConv,

  -- For component ADC_interface
  -- CONV          => CONV,
  -- SCK           => SCK,

  -- For component ADC_interface2
  CS_N          => CS_N,
  DCLOCK        => DCLOCK,

  X_Shift_en    => X_Shift_en,
  Y_Shift_en    => Y_Shift_en,
  Z_Shift_en    => Z_Shift_en,

  -- Signals that new data has been received from ADC
  new_data      => new_data
```

```
  );

  X_SHIFT16: shift16
  port map
  (
    -- Control inputs
    a_rst    => a_rst,
    mclk     => DCLOCK,
    -- mclk     => mclk,
    enable   => X_Shift_en,

    -- 1-bit data input
    indata   => X_SDO,

    -- 16-bit data output
    outdata => X_Data_raw
  );
  Y_SHIFT16: shift16
  port map
  (
    -- Control inputs
    a_rst    => a_rst,
    mclk     => DCLOCK,
    -- mclk     => mclk,

    enable   => Y_Shift_en,

    -- 1-bit data input
    indata   => Y_SDO,

    -- 16-bit data output
    outdata => Y_Data_raw
  );
  Z_SHIFT16: shift16
  port map
  (
    -- Control inputs
    a_rst    => a_rst,
    mclk     => DCLOCK,
    -- mclk     => mclk,

    enable   => Z_Shift_en,

    -- 1-bit data input
    indata   => Z_SDO,

    -- 16-bit data output
    outdata => Z_Data_raw
  );

  UART_TRANSMITTER_UNIT: UART_Transmitter
  port map
  (
    -- Control inputs
    UART_enable    => UART_enable,
    a_rst          => a_rst,
    s_rst          => s_rst,
    mclk           => mclk,
    new_data       => new_data,

    -- Control data
    Outdata_X      => X_Data,
    Outdata_Y      => Y_Data,
    Outdata_Z      => Z_Data,

    -- Data output
    AUX_DATA_OUT => AUX_DATA_OUT
  );
```

```
    PCM_INTERFACE_UNIT: PCM_interface
    port map
    (
      -- Control inputs
      PCM_enable     => PCM_enable,
      a_rst          => a_rst,
      s_rst          => s_rst,
      mclk           => mclk,
      SCLK_N         => SCLK_N,
      GATE           => GATE,
      current_axis   => current_axis,

      -- Control data
      Outdata_X      => X_Data,
      Outdata_Y      => Y_Data,
      Outdata_Z      => Z_Data,

      -- Data output
      DATA => DATA
    );

    CRU_CONTROL_UNIT: cru_control
    port map
    (
      SCLK_N  => SCLK_N,
      MINF    => MINF,
      MAJF    => MAJF,
      GATE    => GATE,
      JP1     => JP1,
      JP2     => JP2,

      -- Clock and reset signals
      a_rst => a_rst,
      s_rst => s_rst,
      mclk  => mclk,

      -- Communication signals
      UART_enable   => UART_enable,
      PCM_enable    => PCM_enable,
      current_axis  => current_axis,

      -- Magnetometer and ADC control
      Vsr             => Vsr,
      polarity_demod  => polarity_demod,
      StartConv       => StartConv,

      -- Test signals
      Test_vector     => Test_vector
    );

end architecture top;
```

code/SRADS_MAG.vhd

C.2 Clock Reset and Control Unit

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity cru_control is
  port
  (
    -- External signals
    SCLK_N     : in std_logic;
    MINF       : in std_logic;
    MAJF       : in std_logic;
    GATE       : in std_logic;
    JP1        : in std_logic;
    JP2        : in std_logic;

    -- Clock and reset signals
    a_rst      : buffer std_logic;
    s_rst      : buffer std_logic;
    mclk       : buffer std_logic;

    -- Communication signals
    UART_enable      : buffer std_logic;
    PCM_enable       : buffer std_logic;
    current_axis     : out integer range 0 to 2;

    -- Magnetometer and ADC control
    Vsr              : buffer std_logic;
    polarity_demod   : buffer std_logic;
    StartConv        : buffer std_logic;

    Test_vector      : out std_logic_vector(3 downto 0)
  );
end entity cru_control;

architecture rtl of cru_control is

  component osc_altufm_osc_rv5 is
    port
    (
      osc     : out std_logic;
      oscena  : in  std_logic
    );
  end component osc_altufm_osc_rv5;
  component rst_unit is
    port
    (
      mclk : in std_logic;
      rst  : out std_logic
    );
  end component rst_unit;

  signal jps                   : std_logic_vector(1 downto 0);
  signal PCM_main              : std_logic;
  signal StartConv_noPCM       : std_logic;
  signal StartConv_PCM         : std_logic;
  signal Vsr_PCM               : std_logic;
  signal Vsr_noPCM             : std_logic;
  -- signal polarity_demod_PCM   : std_logic;
  -- signal polarity_demod_noPCM : std_logic;
  signal polarity_demod_inv    : std_logic;
  signal Word_count            : integer range 0 to 144;
  signal MINF_count            : integer range 0 to 64;
  signal sreg                  : std_logic_vector(15 downto 0);

begin
```

```
--**************************
--********* BEGIN ***********
--**************************
  jps <= (not JP2) & (not JP1);

  s_rst <= MAJF;

  with PCM_enable select
    StartConv <= StartConv_PCM     when '1',
                 StartConv_noPCM   when others;

  with PCM_enable select
    Vsr <= Vsr_PCM     when '1',
           Vsr_noPCM   when others;

  -- with PCM_enable select
    -- polarity_demod <= polarity_demod_PCM    when '1',
                      -- polarity_demod_noPCM when others;

  POLARITY_DEMOD_DELAY:
  process(mclk, Vsr)
  -- Delay demodulation in relation to Vsr (by delaycycles * mclk period ~200ns)
  constant delaycycles  : integer := 2;
  begin
  if rising_edge(mclk) then
    sreg(0) <= Vsr;
    for i in 0 to delaycycles-1 loop
      sreg(i+1) <= sreg(i);
    end loop;
    if polarity_demod_inv = '1' then
      polarity_demod <= not sreg(delaycycles);
    else
      polarity_demod <= sreg(delaycycles);
    end if;
  end if;
  end process POLARITY_DEMOD_DELAY;


  -- The Frame counter counts minor frames from 0-2.
  -- Every 3rd minor frame the counter resets.
  -- The axis selector chooses the active axis depending on this counter.
  -- It is also reset with MAJF (every 64 MINF) so as 64 is not divisible by 3,
  -- the X-axis-measurement will be sent an extra time every MAJF.
  FRAME_COUNTER:
  process(SCLK_N, a_rst, MINF, MAJF)
  begin
    if a_rst = '1' then
      MINF_count <= 0;
    elsif falling_edge(SCLK_N) then
      if MAJF = '1' then -- is this OK? SYNCH?
        MINF_count <= 0;
      elsif MINF = '1' then
        if MINF_count = 2 then -- is this OK? SYNCH?
          MINF_count <= 0;
        else
          MINF_count <= MINF_count + 1;
        end if;
      end if;
    end if;
  end process FRAME_COUNTER;

  WORD_COUNTER:
  process(SCLK_N, a_rst, MINF)
    variable bits : integer range 0 to 7;
    constant bits_max : integer := 7;
  begin
    if a_rst = '1' then
      Word_count <= 0;
```

```
      bits := 0;
    elsif falling_edge(SCLK_N) then
      if MINF = '1' then -- is this OK? SYNCH?
        Word_count <= 0;
        bits := 0;
      else
        if bits = bits_max then
          bits := 0;
          Word_count <= Word_count+1;
        else
          bits := bits + 1;
        end if;
      end if;
    end if;
  end process WORD_COUNTER;

  -- Drives the 'StartConv_PCM' signal. It should drive a clock cycle long pulse
  -- when ADC conversion is supposed to start.
  -- This will be different for the two cards:
  -- 1. Card #1 will be connected to the TX1-main encoder and will sample
  --    at the start of every minor frame (2893,5Hz)
  -- 2. Card #2 will be connected to the TX2-slave encoder and will sample
  --    at the start of every third minor frame (964,5Hz)
  -- Update: Card #2 also got three full slots per minor frame and samples
  --         at the start of every minor frame
  -- Also drives 'Vsr_PCM' and 'polarity_demod_PCM' signals
  MAG_ADC_TIMER_PCM:
  process(a_rst, mclk, Word_count)
    constant word_max                 : integer := 144;
    constant word_StartConv           : integer := 2;
    constant word_vsr_on              : integer := word_max / 4 + word_StartConv;
    constant word_vsr_off             : integer := (3*word_max) / 4 + word_StartConv;
-- constant word_polarity_demod_on   : integer := (3*word_max) / 4 + word_StartConv;
-- constant word_polarity_demod_off  : integer := word_max / 4 + word_StartConv;
  begin
    -- StartConv_PCM       <= '0';
    -- Vsr_PCM             <= '0';
    -- polarity_demod_PCM  <= '0';
    if a_rst = '1' then
      StartConv_PCM <= '0';
      Vsr_PCM <= '0';
      -- polarity_demod_PCM <= '0';
    elsif rising_edge(mclk) then
      StartConv_PCM <= '0';
      if Word_count = word_StartConv then
        StartConv_PCM <= '1';
      end if;
      if Word_count = word_vsr_on then
        Vsr_PCM <= '1';
      end if;
      if Word_count = word_vsr_off then
        Vsr_PCM <= '0';
      end if;

      -- Replaced by a process that drives polarity_demod by delaying Vsr
      -- if Word_count = word_polarity_demod_on then
        -- polarity_demod_PCM <= '1';
      -- end if;
      -- if Word_count = word_polarity_demod_off then
        -- polarity_demod_PCM <= '0';
      -- end if;
    end if;
  end process MAG_ADC_TIMER_PCM;

  -- Drives the 'StartConv_noPCM' signal. It should give a clock cycle long
  -- pulse when ADC conversion is supposed to start.
  -- The CPLD has an internal oscillator of 5Mhz. Sample rate is deduced by
  -- that clock to get as close to 2893,5 Hz as possible. 1728*200ns = 345.6 us
```

```
  -- Also drives 'Vsr_PCM' and 'polarity_demod_PCM' signals
MAG_ADC_TIMER_NOPCM:
process(mclk, a_rst)
  variable counter : integer range 0 to 1728 := 0;
  constant count_max                : integer := 1728;
  constant count_StartConv          : integer := 2;
  constant count_vsr_on             : integer := count_max / 4;
  constant count_vsr_off            : integer := (3*count_max) / 4;
  -- constant count_polarity_demod_on  : integer := (3*count_max) / 4;
  -- constant count_polarity_demod_off : integer := count_max / 4;
begin
  if a_rst = '1' then
    counter := 0;
    StartConv_noPCM <= '0';
    Vsr_noPCM <= '0';
    -- polarity_demod_noPCM <= '0';
  elsif rising_edge(mclk) then
    counter := counter+1;
    StartConv_noPCM <= '0';
    if counter = count_max then
      counter := 0;
    end if;
    if counter = count_StartConv then
      StartConv_noPCM <= '1';
    end if;
    if counter = count_vsr_on then
      Vsr_noPCM <= '1';
    end if;
    if counter = count_vsr_off then
      Vsr_noPCM <= '0';
    end if;

    -- Replaced by a process that drives polarity_demod by delaying Vsr
    -- if counter = count_polarity_demod_on then
      -- polarity_demod_noPCM <= '1';
    -- end if;
    -- if counter = count_polarity_demod_off then
      -- polarity_demod_noPCM <= '0';
    -- end if;
  end if;
end process MAG_ADC_TIMER_NOPCM;

Test_vector(0) <= Vsr;
Test_vector(1) <= '0';
Test_vector(2) <= '0';
Test_vector(3) <= '0';

JUMPER_MODE_SELECTOR:
process(mclk, jps, StartConv, Vsr, polarity_demod)
begin
  polarity_demod_inv <= '0';
  PCM_main <= '1';
  UART_enable <= '0';
  PCM_enable <= '1';
  case jps is
    when "00" =>
    when "01" =>
      polarity_demod_inv <= '1';
    when "10" =>
      PCM_main <= '0';
    when "11" =>
      PCM_main <= '0';
      polarity_demod_inv <= '1';
    when others =>
  end case;
end process JUMPER_MODE_SELECTOR;

AXIS_SELECTOR:
```

```
  process(PCM_main, MINF_count, Word_count)
  begin
    if (PCM_main = '1') then
      -- TX1 format
      -- X-data should be ready by word 36 (using start of frame)
      -- Y-data should be ready by word 84 (using 60)
      -- Z-data should be ready by word 132 (using 80)
      if (Word_count <= 40) then
        current_axis <= 0;
      elsif (Word_count <= 90) then
        current_axis <= 1;
      elsif (Word_count > 90) then
        current_axis <= 2;
      else
        current_axis <= 0;
      end if;
    else
      -- TX2 format
      -- X-data should be ready by word 44 (using start of frame)
      -- Y-data should be ready by word 68 (using 60)
      -- Z-data should be ready by word 92 (using 80)
      if (Word_count <= 60) then
        current_axis <= 0;
      elsif (Word_count <= 80) then
        current_axis <= 1;
      elsif (Word_count > 80) then
        current_axis <= 2;
      else
        current_axis <= 0;
      end if;
    end if;
  end process AXIS_SELECTOR;

  OSC_5MHZ: osc_altufm_osc_rv5
  port map
  (
    osc => mclk,
    oscena => '1'
  );

  RESET_UNIT: rst_unit
  port map
  (
    mclk => mclk,
    rst => a_rst
  );

end architecture rtl;
```

code/cru_control.vhd

C.3 Reset Unit

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity rst_unit is
  port
  (
    mclk : in std_logic;
    rst : out std_logic
  );
end entity rst_unit;

architecture rtl of rst_unit is

  -- powerup state machine types and signals
  type powerup_states is (powerup_st0, powerup_st1, idle_st);
  signal powerup_st : powerup_states;

  begin

  -- Controls the powerup of the CPLD and internal reset signal
  -- COMB controls outputs from states
  -- SEQ controls the sequential logic
  POWERUP_STATE_COMB: process(powerup_st)
  begin
    case powerup_st is
    -- The two powerup states output rst signal high
      when powerup_st0 =>
        rst <= '1';
      when powerup_st1 =>
        rst <= '1';
    -- This is the default state between measurements
      when idle_st =>
        rst <= '0';
      when others =>
        rst <= '1';
    end case;
  end process POWERUP_STATE_COMB;
  POWERUP_STATE_SEQ: process(mclk)
  begin
    if rising_edge(mclk) then
      case powerup_st is
        -- According to the CPLD datasheet,
        -- the device will powerup into this state
        when powerup_st0 =>
          powerup_st <= powerup_st1;
        -- A second powerup-state ensures a minimum of
        -- one clock period of power_up time
        when powerup_st1 =>
          powerup_st <= idle_st;
        -- After powerup, the device will enter this state
        -- and stay here
        when idle_st =>
          powerup_st <= idle_st;
        -- In case of undefined state,
        -- the device goes into one powerup-state
        when others =>
          powerup_st <= powerup_st1;
      end case;
    end if;
  end process POWERUP_STATE_SEQ;

end architecture rtl;
```

code/rst_unit.vhd

C.4 Oscillator Block

```
-- megafunction wizard: %MAX II oscillator%
-- GENERATION: STANDARD
-- VERSION: WM1.0
-- MODULE: altufm_osc

-- ============================================================
-- File Name: osc.vhd
-- Megafunction Name(s):
--       altufm_osc
--
-- Simulation Library Files(s):
--       maxii
-- ============================================================
-- ************************************************************
-- THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
--
-- 9.1 Build 222 10/21/2009 SJ Web Edition
-- ************************************************************


--Copyright (C) 1991-2009 Altera Corporation
--Your use of Altera Corporation's design tools, logic functions
--and other software and tools, and its AMPP partner logic
--functions, and any output files from any of the foregoing
--(including device programming or simulation files), and any
--associated documentation or information are expressly subject
--to the terms and conditions of the Altera Program License
--Subscription Agreement, Altera MegaCore Function License
--Agreement, or other applicable license agreement, including,
--without limitation, that your use is for the sole purpose of
--programming logic devices manufactured by Altera and sold by
--Altera or its authorized distributors.  Please refer to the
--applicable agreement for further details.


--altufm_osc CBX_AUTO_BLACKBOX="ALL" OSC_FREQUENCY=180000 osc oscena
--VERSION_BEGIN 9.1 cbx_altufm_osc 2009:10:21:21:22:16:SJ cbx_maxii
    2009:10:21:21:22:16:SJ cbx_mgl 2009:10:21:21:37:49:SJ cbx_stratixii
    2009:10:21:21:22:16:SJ cbx_util_mgl 2009:10:21:21:22:16:SJ  VERSION_END

LIBRARY maxii;
USE maxii.all;

--synthesis_resources = maxii_ufm 1
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY  osc_altufm_osc_rv5 IS
  PORT
  (
    osc   : OUT  STD_LOGIC;
    oscena : IN  STD_LOGIC
  );
END osc_altufm_osc_rv5;

ARCHITECTURE RTL OF osc_altufm_osc_rv5 IS

  SIGNAL  wire_gnd : STD_LOGIC;
  SIGNAL  wire_vcc : STD_LOGIC;
  SIGNAL  wire_maxii_ufm_block1_osc  : STD_LOGIC;
  COMPONENT  maxii_ufm
  GENERIC
  (
    ADDRESS_WIDTH : NATURAL := 9;
    ERASE_TIME  : NATURAL := 500000000;
```

```
    INIT_FILE : STRING := "UNUSED";
    OSC_SIM_SETTING : NATURAL := 180000;
    PROGRAM_TIME : NATURAL := 1600000;
    lpm_type : STRING := "maxii_ufm"
  );
  PORT
  (
    arclk : IN STD_LOGIC := '0';
    ardin : IN STD_LOGIC := '0';
    arshft : IN STD_LOGIC := '1';
    bgpbusy : OUT STD_LOGIC;
    busy : OUT STD_LOGIC;
    drclk : IN STD_LOGIC := '0';
    drdin : IN STD_LOGIC := '0';
    drdout : OUT STD_LOGIC;
    drshft : IN STD_LOGIC := '1';
    erase : IN STD_LOGIC := '0';
    osc : OUT STD_LOGIC;
    oscena : IN STD_LOGIC := '0';
    program : IN STD_LOGIC := '0'
  );
  END COMPONENT;
BEGIN

  wire_gnd <= '0';
  wire_vcc <= '1';
  osc <= wire_maxii_ufm_block1_osc;
  maxii_ufm_block1 : maxii_ufm
    GENERIC MAP (
    ADDRESS_WIDTH => 9,
    OSC_SIM_SETTING => 180000
    )
    PORT MAP (
    arclk => wire_gnd,
    ardin => wire_gnd,
    arshft => wire_gnd,
    drclk => wire_gnd,
    drdin => wire_gnd,
    drshft => wire_vcc,
    osc => wire_maxii_ufm_block1_osc,
    oscena => oscena
    );

END RTL; --osc_altufm_osc_rv5
--VALID FILE


LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY osc IS
  PORT
  (
    oscena    : IN STD_LOGIC ;
    osc    : OUT STD_LOGIC
  );
END osc;


ARCHITECTURE RTL OF osc IS

  SIGNAL sub_wire0  : STD_LOGIC ;



  COMPONENT osc_altufm_osc_rv5
  PORT (
      oscena  : IN STD_LOGIC ;
```

```
        osc : OUT STD_LOGIC
    );
    END COMPONENT;

BEGIN
    osc     <= sub_wire0;

    osc_altufm_osc_rv5_component : osc_altufm_osc_rv5
    PORT MAP (
      oscena => oscena,
      osc => sub_wire0
    );



END RTL;

-- ============================================================
-- CNX file retrieval info
-- ============================================================
-- Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING "MAX II"
-- Retrieval info: PRIVATE: INTENDED_DEVICE_PART STRING ""
-- Retrieval info: PRIVATE: INTERFACE_CHOICE NUMERIC "4"
-- Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING "1"
-- Retrieval info: PRIVATE: VERSION_NUMBER NUMERIC "0"
-- Retrieval info: CONSTANT: OSC_FREQUENCY NUMERIC "180000"
-- Retrieval info: USED_PORT: osc 0 0 0 0 OUTPUT NODEFVAL osc
-- Retrieval info: USED_PORT: oscena 0 0 0 0 INPUT NODEFVAL oscena
-- Retrieval info: CONNECT: @oscena 0 0 0 0 oscena 0 0 0 0
-- Retrieval info: CONNECT: osc 0 0 0 0 @osc 0 0 0 0
-- Retrieval info: GEN_FILE: TYPE_NORMAL osc.vhd TRUE
-- Retrieval info: GEN_FILE: TYPE_NORMAL osc.inc TRUE
-- Retrieval info: GEN_FILE: TYPE_NORMAL osc.cmp TRUE
-- Retrieval info: GEN_FILE: TYPE_NORMAL osc.bsf TRUE
-- Retrieval info: GEN_FILE: TYPE_NORMAL osc_inst.vhd TRUE
-- Retrieval info: GEN_FILE: TYPE_NORMAL osc_syn.v TRUE
-- Retrieval info: LIB_FILE: maxii
```

code/osc.vhd

C.5 ADC interface (LTC1864)

```
-- Interface to the LTC 1864L uPower ADC, customized
-- for the SRADS_MAG and ICI-3 PCM encoder
-- SCK maximum 8Mhz, mclk is max 5.5Mhz, can be used directly
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ADC_interface is
  port (
    -- ADC control input
    a_rst      : in std_logic;
    s_rst      : in std_logic;
    mclk       : in std_logic;
    StartConv  : in std_logic;

    -- ADC control
    CONV   : out std_logic;
    SCK    : out std_logic;

    -- Shiftreg control
    X_Shift_en   : out std_logic;
    Y_Shift_en   : out std_logic;
    Z_Shift_en   : out std_logic;

    -- UART control
    new_data     : out std_logic
  );
end entity ADC_interface;

architecture rtl of ADC_interface is

  type adc_states is (idle_st, conv_st, data_receive_st);
  signal adc_st       : adc_states;
  signal sck_en       : std_logic;
  signal rst          : std_logic;
  signal clk          : std_logic;

begin

  X_Shift_en  <= sck_en;
  Y_Shift_en  <= sck_en;
  Z_Shift_en  <= sck_en;
  SCK <= mclk when sck_en = '1' else 'Z';

  -- Controls the states of the ADC interface
  -- COMB controls outputs from states
  -- SEQ controls the sequential logic
  ADC_STATE_COMB: process(adc_st)
  begin
    case adc_st is
      when idle_st =>
      -- This is the inactive state between measurements
        CONV <= '0';
        sck_en <= '0';
      when conv_st =>
      -- Initiate conversion and keep signal high until conversion done
        CONV <= '1';
        sck_en <= '0';
      when data_receive_st =>
      -- Enable serial clock to receive data
        CONV <= '0';
        sck_en <= '1';
      when others =>
      -- in case of undefined state all outputs low
        CONV <= '0';
```

```
        sck_en <= '0';
    end case;
  end process ADC_STATE_COMB;

  ADC_STATE_SEQ: process(mclk, a_rst)
  -- DETERMINING ADC TIMING
  -- t_conv(max) = 4.66us ~= 5us
  -- mclk frequency is max 5.56 ~= 6Mhz
  -- minimum period is 1/6Mhz = 167ns
  -- 5000 ns / 167 ns = 29,9. Using t_conv_max = 32.
  -- t_conv = 32 * SCK
  -- t_SMPL = 16 * SCK
  -- t_CYC = t_conv + t_SMPL = 48 * SCK
  -- Minimum waiting time calculation (using 6Mhz as reference, actually 5.56Mhz)
  -- Min t_conv time = 32 * 167ns = 5.3us
  -- Min total cycle time = 48 * 167 ~= 8us
  -- Maximum waiting time calculation (using 3Mhz as reference, actually 3.3Mhz)
  -- Max t_conv time = 32 * 333ns = 10.7us
  -- Max total cycle time = 48 * 333 ~= 16 us
  -- t_conv using t_conv_max = 32 is guaranteed to be between 5.3us - 10.7us
  -- total cycle time before data is ready is guaranteed to be between 8us - 16us
  -- Sampling will start inside word 1 in the frame.
  -- Requirement for data to be ready is before word 36 (TX1-main)
  -- Calculating using word 30 = bit 30*8 = bit 240
  -- SCLK period is 300ns, which means that the first data must be ready
  -- after 240*300ns ~= 72us
  -- Even with exaggerated worst-case timings, there's plenty of time.
    constant t_conv_max     : integer := 32;
    variable t_conv_count   : integer range 0 to 32 := 0;
    variable bits_received  : unsigned (3 downto 0) := "0000";
  begin
    if a_rst = '1' then
      bits_received := (others => '0');
      t_conv_count := 0;
      adc_st <= idle_st;
    elsif rising_edge(mclk) then
      new_data <= '0';
      case adc_st is
        when idle_st =>
        -- This state waits for StartConv signal, then goes to conv_st
          t_conv_count := 0;
          bits_received := "0000";
          if StartConv = '1' then
            adc_st <= conv_st;
          end if;
        when conv_st =>
        -- Starts conversion and waits for conversion time t_conv, goes to
            data_receive
          if t_conv_count = t_conv_max then
            adc_st <= data_receive_st;
          end if;
          t_conv_count := t_conv_count + 1;
        when data_receive_st =>
        -- Waits for 16 clock cycles to receive 16 bits
          if bits_received = "1111" then
            adc_st <= idle_st;
            new_data <= '1';
          end if;
          bits_received := bits_received + 1;
        when others =>
          adc_st <= idle_st;
      end case;
    end if;
  end process ADC_STATE_SEQ;
end architecture rtl;
```

code/ADC_interface.vhd

C.6 ADC interface 2(AD7684)

```
-- Interface to the AD7684 16-bit 100kSPS ADC, customized for
-- the SRADS_MAG for IDI-3 and max 3kHz sampling rate
-- SCK maximum 8Mhz, mclk is max 5.5Mhz, can be used directly
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ADC_interface2 is
  port (
    -- ADC control input
    a_rst      : in std_logic;
    s_rst      : in std_logic;
    mclk       : in std_logic;
    StartConv  : in std_logic;

    -- ADC control
    CS_N       : out std_logic;
    DCLOCK     : buffer std_logic;

    -- Shiftreg control
    X_Shift_en  : out std_logic;
    Y_Shift_en  : out std_logic;
    Z_Shift_en  : out std_logic;

    -- UART control
    new_data    : out std_logic
  );
end entity ADC_interface2;

architecture rtl of ADC_interface2 is

  signal data_en  : std_logic;

begin

  X_Shift_en  <= data_en;
  Y_Shift_en  <= data_en;
  Z_Shift_en  <= data_en;

  -- DETERMINING ADC TIMING
  -- Maximum DCLOCK frequency for AD7684 is 2,9Mhz, so we'll use
  -- an adc clock of half the frequency of mclk
  -- Max frequency of mclk = 5,5Mhz, which means we'll use
  -- an adc clock of max. 2,75Mhz.
  -- When in idle state StartConv signal might be signalled
  -- either in rising or falling edge of the adc_clock
  -- To adhere to the timing t_CSD = 0us
  -- Case rising: Will have to wait one cycle of mclk
  --              to get DCLOCK low, maximum 1/3.3Mhz = 300ns
  -- Case falling: DCLOCK is already low
  -- To adhere to the timing t_SUCS = minimum 20ns
  -- Case rising: will take 600ns before another rising edge
  -- Case falling: will take 300ns before rising edge

  -- We also have to guarantee that conversions will not be started
  -- before current cycle is done.
  -- One complete cycle before the ADC is ready again will take
  -- a maximum of 24 cycles of the adc clock plus t_acq which is 400ns.
  -- adc clock is mclk/2 which means a minimum
  -- of 3.3Mhz/2 = 1.65Mhz which yields a period of ~606ns.
  -- Maximum total cycle time will therefore be ~24*600ns + 400ns = 14,8 us.
  -- Sampling rate is ~3kHz which yields 333us, so cycle time is
  -- well within limits of expected operation.
  ADC_CONTROL:
  process(mclk, a_rst, s_rst)
```

```
    type ADC_states is (idle_st, conv_st, pre_data_st, data_st, post_data_st,
        new_data_signal);
67  variable ADC_state    : ADC_states := idle_st;
    variable dclock_count : integer range 0 to 16 := 0;
    constant data_start   : integer := 6;
    constant data_maxbit  : integer := 16;
71  variable adc_clock    : std_logic := '0';
  begin
    if a_rst = '1' then
      adc_clock := '0';
75    dclock_count := 0;
      ADC_state := idle_st;
    elsif rising_edge(mclk) then
      if s_rst = '1' then
79      adc_clock := '0';
        dclock_count := 0;
        ADC_state := idle_st;
      else
83      adc_clock := not adc_clock;
        case ADC_state is
          when idle_st =>
            dclock_count := 0;
87          if StartConv = '1' then
              ADC_state := conv_st;
            end if;
          when conv_st =>
91          if adc_clock = '1' then     -- if rising_edge(adc_clock)
              dclock_count := dclock_count + 1;
              if dclock_count = data_start then
                dclock_count := 0;
95              ADC_state := pre_data_st;
              end if;
            end if;
          when pre_data_st =>
99          if adc_clock = '0' then  -- if falling_edge(adc_clock)
              ADC_state := data_st;
            end if;
          when data_st =>
103         if adc_clock = '0' then     -- if falling_edge(adc_clock)
              dclock_count := dclock_count + 1;
              if dclock_count = data_maxbit then
                ADC_state := new_data_signal;
107           end if;
            end if;
          when new_data_signal =>
            ADC_state := idle_st;
111       when others =>
            ADC_state := idle_st; -- MAPL added 22.08.2011 Undefined failsafe
        end case;
      end if;
115 end if;
    -- Combinatorial section
    DCLOCK <= adc_clock;
    case ADC_state is
119   when idle_st =>
      -- This is the "inactive" state
        CS_N <= '1';
        data_en <= '0';
123     new_data <= '0';
      when conv_st =>
      -- Initiate conversion by pulling CS_N low,
      -- then wait for 6 rising edges of adc_clock
127     CS_N <= '0';
        data_en <= '0';
        new_data <= '0';
      when pre_data_st =>
131   -- Wait an additional half adc_clock cycle to account for DCLOCK skew
      -- From now on everything happens on falling edge of DCLOCK
```

```
      -- so the shift register can process data on rising edge
        CS_N <= '0';
        data_en <= '0';
        new_data <= '0';
      when data_st =>
      -- Enable shift registers to receive data
        CS_N <= '0';
        data_en <= '1';
        new_data <= '0';
      when new_data_signal =>
      -- Pull CS_N high and give a one mclk cycle pulse on new_data
        CS_N <= '1';
        data_en <= '0';
        new_data <= '1';
      when others =>
      -- in case of undefined state equal idle_st
        CS_N <= '1';
        data_en <= '0';
        new_data <= '0';
      end case;
  end process ADC_CONTROL;
end architecture rtl;
```

code/ADC_interface2.vhd

C.7 16-bit Shift Register

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shift16 is
  port (
    -- Control inputs
    a_rst    : in std_logic;
    mclk     : in std_logic;
    enable   : in std_logic;

    -- 1-bit data input
    indata   : in std_logic;

    -- 16-bit data output
    outdata : out std_logic_vector(15 downto 0)
  );
end entity shift16;

architecture rtl of shift16 is
  signal outdata_i : std_logic_vector(15 downto 0);

  begin

  outdata <= outdata_i;

  SHIFT_REG: process(mclk, a_rst, enable)
  begin
    if a_rst = '1' then
      outdata_i <= (others => '0');
    -- elsif falling_edge(mclk) and enable = '1' then
    elsif rising_edge(mclk) and enable = '1' then
      outdata_i(0) <= indata;
      for i in 1 to 15 loop
        outdata_i(i) <= outdata_i(i-1);
      end loop;
    end if;
  end process SHIFT_REG;
end architecture rtl;
```

code/shift16.vhd

C.8 UART Transmitter

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity UART_Transmitter is
  port
  (
    -- Control inputs
    UART_enable   : in std_logic;
    a_rst         : in std_logic;
    s_rst         : in std_logic; -- For possible future use
    mclk          : in std_logic;
    new_data      : in std_logic;

    -- Control data
    Outdata_X     : in std_logic_vector(15 downto 0);
    Outdata_Y     : in std_logic_vector(15 downto 0);
    Outdata_Z     : in std_logic_vector(15 downto 0);

    -- Data output
    AUX_DATA_OUT  : out std_logic
  );
end entity UART_Transmitter;

architecture rtl of UART_Transmitter is

  signal data_x     : std_logic_vector(15 downto 0);
  signal data_y     : std_logic_vector(15 downto 0);
  signal data_z     : std_logic_vector(15 downto 0);
  signal UART_data  : std_logic_vector(7 downto 0);
  signal TX_start   : std_logic;
  signal TX_done    : std_logic;

begin
  data_x <= Outdata_X;
  data_y <= Outdata_Y;
  data_z <= Outdata_Z;

  -- Implementing enable function of the UART Transmitter
  -- holding reset high when not enabled
  -- UART_ENABLER:
  -- process(mclk, UART_enable)
  -- begin
    -- if rising_edge(mclk) then
      -- if UART_enable then
        -- arst_i <= a_rst;
      -- else
        -- arst_i <= '1';
      -- end if;
    -- end if;
  -- end UART_ENABLER;
  -- with UART_enable select
    -- arst_i <= a_rst when '1',
              -- '1'   when others;


  -- Controls the data driver and prepares data to be sent
  TRANSMISSION_CONTROL:
  process (UART_enable, a_rst, mclk, new_data, data_x, data_y, data_z) is
    constant synch1 : std_logic_vector (7 downto 0) := (7 => '0', others => '1');
    constant synch2 : std_logic_vector (7 downto 0) := (others => '1');
    type data_states is (idle_st, send_synch1, send_synch2, sendX_MSB, sendX_LSB,
        sendY_MSB, sendY_LSB, sendZ_MSB, sendZ_LSB);
    --, x_data, y_data, z_data);
    variable data_state : data_states :=  idle_st;
```

```
65  begin
      if a_rst = '1' then
        data_state := idle_st;
        TX_start <= '0';
69    elsif rising_edge(mclk) and UART_enable = '1' then
        TX_start <= '0';
        case data_state is
          when idle_st =>
73          if new_data = '1' then
              data_state := send_synch1;
              TX_start <= '1';
            end if;
77        when send_synch1 =>
            if TX_done = '1' then
              data_state := send_synch2;
              TX_start <= '1';
81          end if;
          when send_synch2 =>
            if TX_done = '1' then
              data_state := sendX_MSB;
85            TX_start <= '1';
            end if;
          when sendX_MSB =>
            if TX_done = '1' then
89            data_state := sendX_LSB;
              TX_start <= '1';
            end if;
          when sendX_LSB =>
93          if TX_done = '1' then
              data_state := sendY_LSB;
              TX_start <= '1';
            end if;
97        when sendY_MSB =>
            if TX_done = '1' then
              data_state := sendX_LSB;
              TX_start <= '1';
101         end if;
          when sendY_LSB =>
            if TX_done = '1' then
              data_state := sendY_LSB;
105           TX_start <= '1';
            end if;
          when sendZ_MSB =>
            if TX_done = '1' then
109           data_state := sendX_LSB;
              TX_start <= '1';
            end if;
          when sendZ_LSB =>
113         if TX_done = '1' then
              data_state := idle_st;
              TX_start <= '1';
            end if;
117       when others =>
            data_state := idle_st;
        end case;
      end if;
121   -- Combinational solution
      case data_state is
        when idle_st =>
          UART_data <= (others => '0');
125     when send_synch1 =>
          UART_data <= synch1;
        when send_synch2 =>
          UART_data <= synch2;
129     when sendX_MSB =>
          UART_data <= data_x(15 downto 8);
        when sendX_LSB =>
          UART_data <= data_x(7 downto 0);
```

```
133       when sendY_MSB =>
            UART_data <= data_y(15 downto 8);
          when sendY_LSB =>
            UART_data <= data_y(7 downto 0);
137       when sendZ_MSB =>
            UART_data <= data_z(15 downto 8);
          when sendZ_LSB =>
            UART_data <= data_z(7 downto 0);
141       when others =>
            UART_data <= (others => '0');
        end case;
    end process TRANSMISSION_CONTROL;
145
    -- TX_Start <= new_data;
    -- UART_data <= Outdata_Y(15 downto 8);

149 -- Sends the 8 bits in Outdata when TX_start goes high
    -- Signals completion of tranmission by driving TX_done high
    UART_DATA_DRIVER:
    process (UART_enable, a_rst, mclk, TX_start, UART_data)
153   type UART_states is (idle_st, start_bit, data_bits, stop_bit);
      variable UART_state   : UART_states := idle_st;
      variable data_bit     : integer range 0 to 7 := 0;
      constant data_bit_max : integer := 7;
157   variable count        : integer range 0 to 10 := 0;
--    constant count_max    : integer := 20; -- Gives UART bitrate of mclk/19
      constant count_max    : integer := 10; -- Gives UART bitrate of mclk/11
                                             -- 4900000/11 = ~445454,5454545455 bps
161 begin
      if a_rst = '1' then
        data_bit := 0;
        count := 0;
165     UART_state := idle_st;
        AUX_DATA_OUT <= '1';
        TX_done <= '0';
      elsif rising_edge(mclk) and UART_enable = '1' then
169     TX_done <= '0';
        AUX_DATA_OUT <= '1';
        case UART_state is
          when idle_st =>
173         if TX_start = '1' then
              data_bit := 0;
              count := 0;
              UART_state := start_bit;
177         end if;
          when start_bit =>
            AUX_DATA_OUT <= '0'; -- Single start bit (logic 0)
            if count = count_max then
181           UART_state := data_bits;
              count := 0;
            else
              count := count+1;
185         end if;
          when data_bits =>
            AUX_DATA_OUT <= UART_data(data_bit);
            if count = count_max then
189           if data_bit = data_bit_max then
                data_bit := 0;
                UART_state := stop_bit;
              else
193             data_bit := data_bit + 1;
              end if;
              count := 0;
            else
197           count := count + 1;
            end if;
          when stop_bit =>
            AUX_DATA_OUT <= '1'; -- Single stop bit (logic 1)
```

```
            if count = count_max then
              UART_state := idle_st;
              TX_done <= '1';
            end if;
            count := count+1;
          when others =>
            UART_state := idle_st;
        end case;
      end if;
      -- Combinatorial part
    end process UART_DATA_DRIVER;

    -- Clock is ~4.9Mhz. Dividing the clock by 22 gives clock frequency of 222,7KHz
    -- The target frequency is 230400Hz, which means that we're ~7700Hz off, which
    -- is a clock deviation of ~3,34%. By some margin, this is acceptable.
    -- However, this code is dependent on knowing with a certain accuracy the
    -- oscillator frequency of the UFM block
    -- UART_CLOCK_GENERATOR:
    -- process (a_rst, mclk)
      -- variable count  : integer  := 0;
    -- begin
      -- if a_rst = '1' then
        -- count := 0;
        -- UART_clk <= '0';
      -- elsif rising_edge(mclk) then
        -- if count = 11 then
          -- UART_clk <= not UART_clk;
          -- count := 0;
        -- end if;
        -- count := count + 1;
      -- end if;
    -- end process UART_CLOCK_GENERATOR;

end architecture rtl;
```

code/UART_Transmitter.vhd

C.9 PCM Interface

```
1  library ieee;
   use ieee.std_logic_1164.all;
   use ieee.numeric_std.all;

5  entity PCM_interface is
     port
     (
       -- Control inputs
9      PCM_enable     : in std_logic;
       a_rst          : in std_logic;
       s_rst          : in std_logic;
       mclk           : in std_logic;
13     SCLK_N         : in std_logic;
       GATE           : in std_logic;
       current_axis   : in integer range 0 to 2;

17     -- Data to send
       OutData_X      : in std_logic_vector(15 downto 0);
       OutData_Y      : in std_logic_vector(15 downto 0);
       OutData_Z      : in std_logic_vector(15 downto 0);
21
       -- Data output
       DATA           : out std_logic
     );
25 end entity PCM_interface;

   architecture rtl of PCM_interface is

29 signal outdata  : std_logic_vector(15 downto 0);
   signal reset    : std_logic;

   begin
33
     with current_axis select
     outdata <=  OutData_X when 0,
                 OutData_Y when 1,
37               OutData_Z when 2;

     GATE_DRIVEN_DATA_SHIFTER:
     process(a_rst, s_rst, SCLK_N, GATE)
41     variable bit_index : integer range 0 to 15 := 15;
     begin
       if a_rst = '1' then
         bit_index := 15;
45       DATA <= '0';
       elsif falling_edge(SCLK_N) then
         if s_rst = '1' then
           bit_index := 15;
49       else
           if gate = '1' then
             DATA <= outdata(bit_index);
             if bit_index = 0 then
53             bit_index := 15;
             else
               bit_index := bit_index - 1;
             end if;
57         end if;
         end if;
       end if;
     end process GATE_DRIVEN_DATA_SHIFTER;
61
   end architecture rtl;
```

code/PCM_interface.vhd

Appendix D

Schematics v1.1

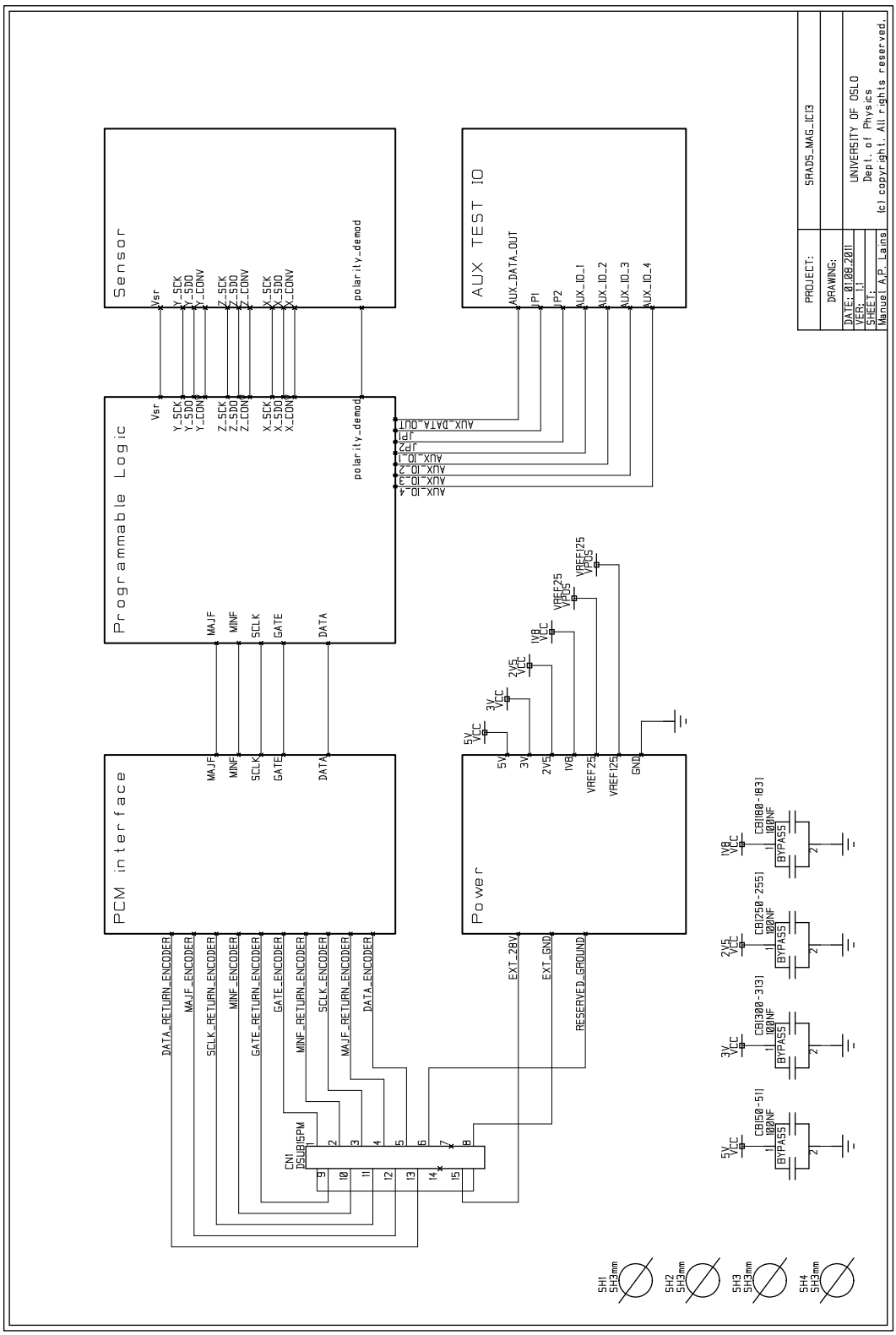

Figure D.1: Top Block

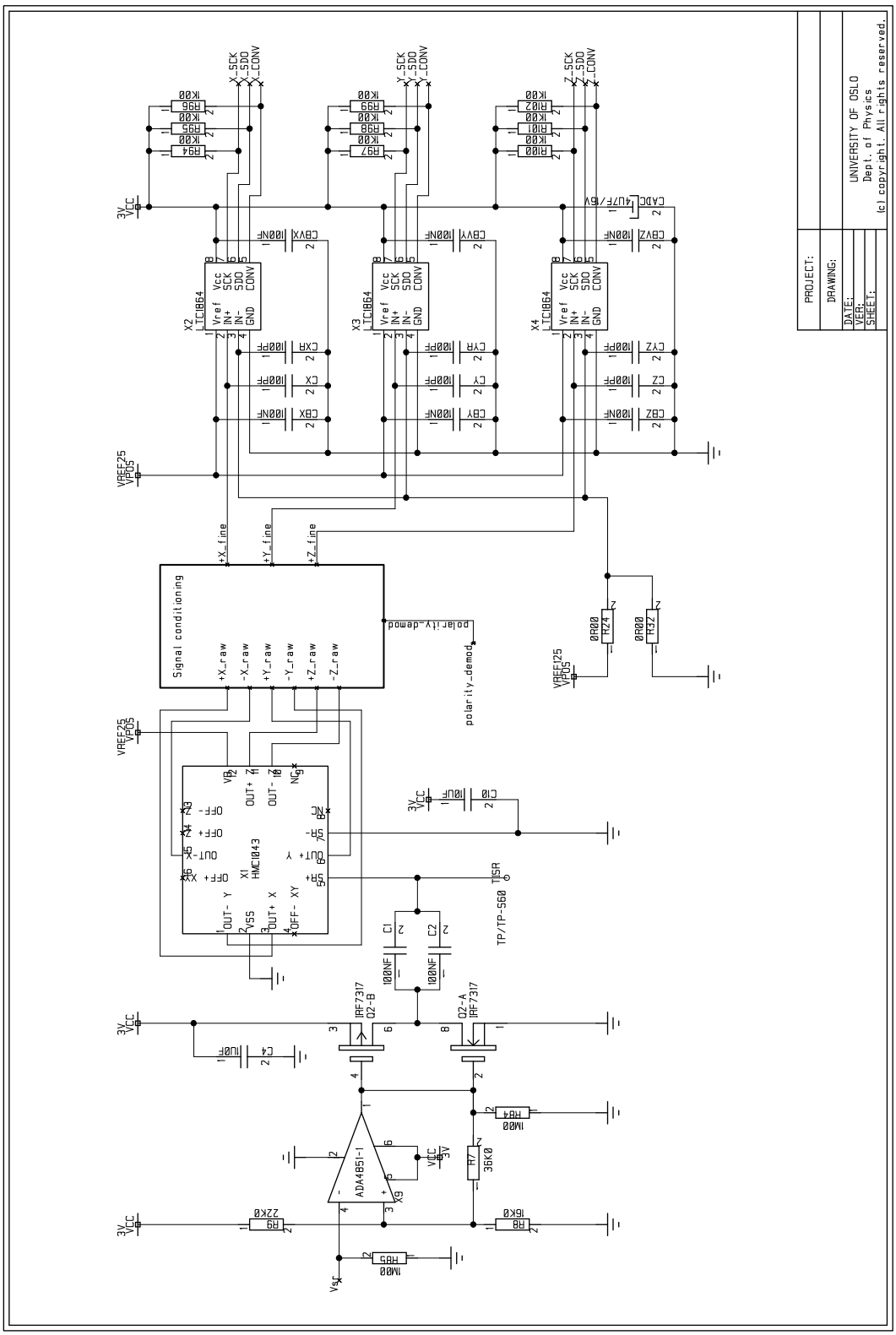

Figure D.2: Sensor Block

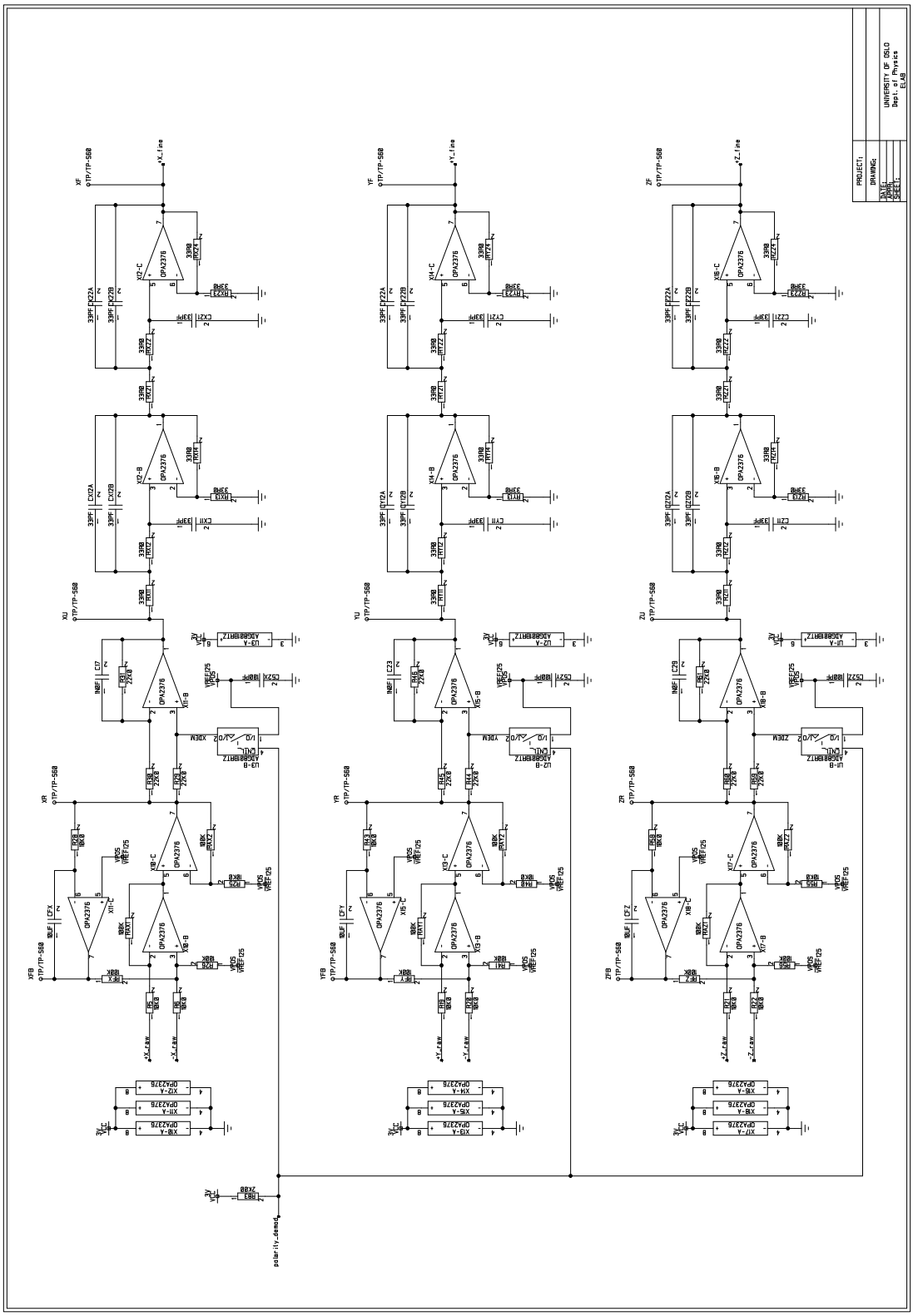

Figure D.3: Signal Conditioning Block

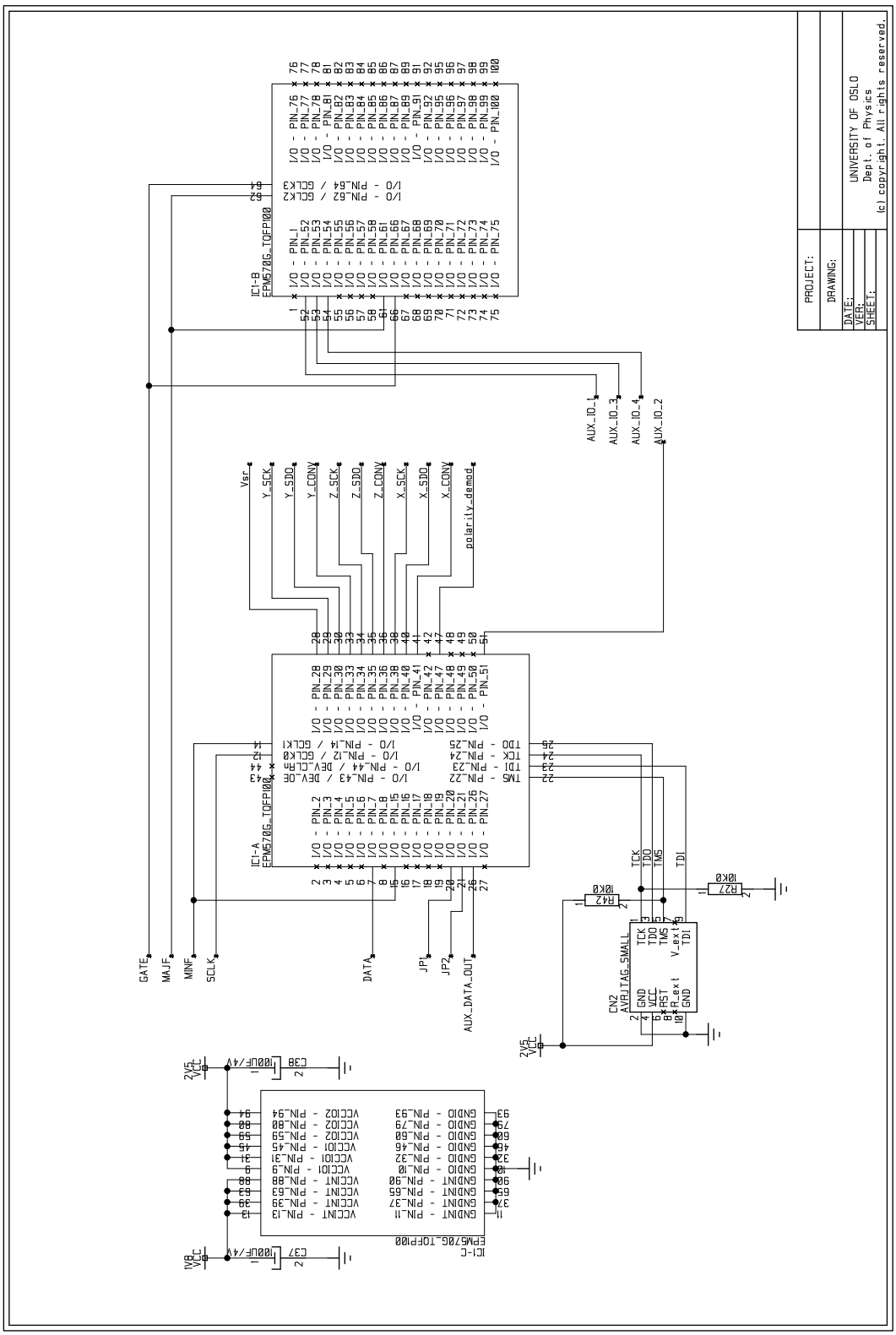

Figure D.4: Programmable Logic Block

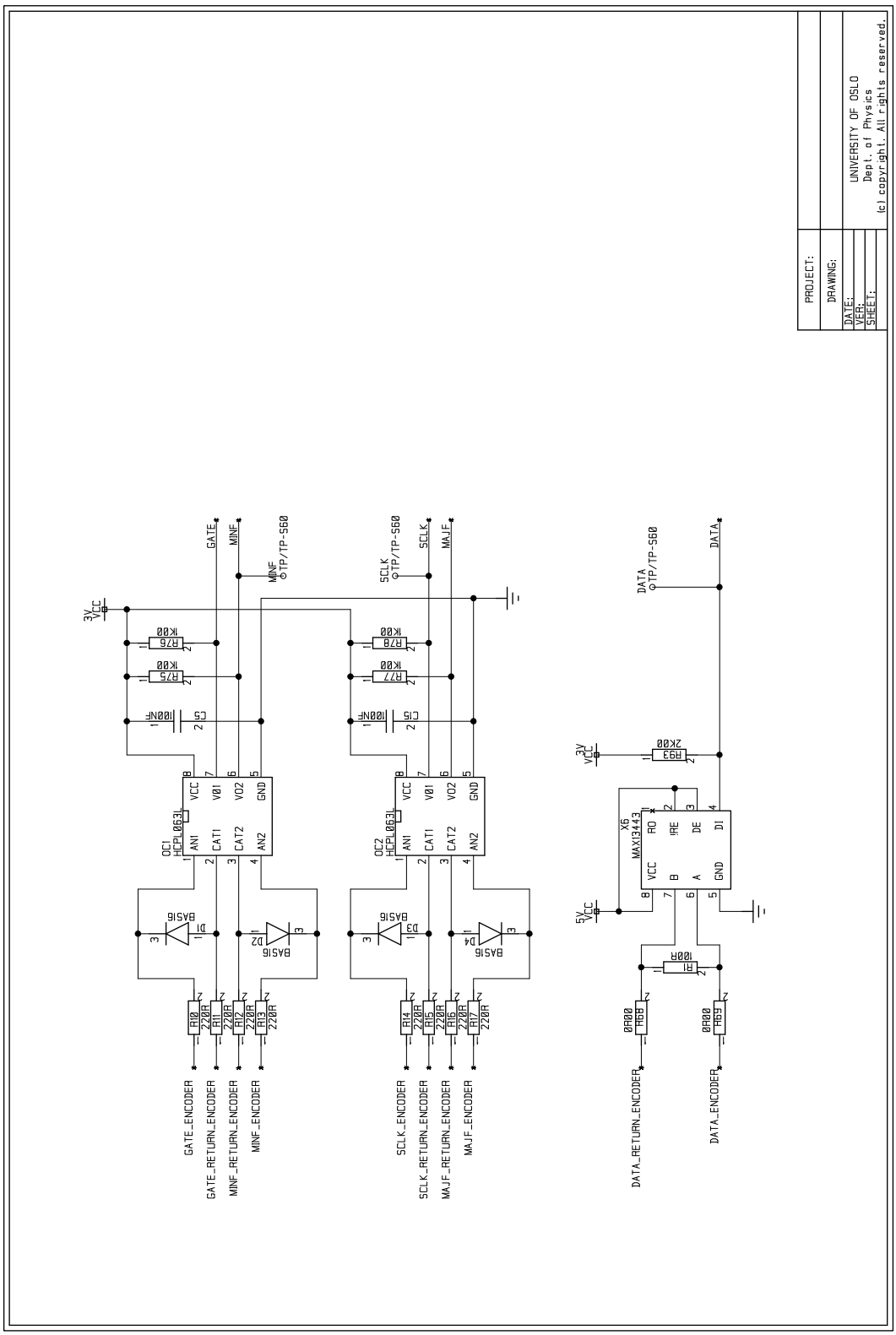

Figure D.5: PCM Interface Block

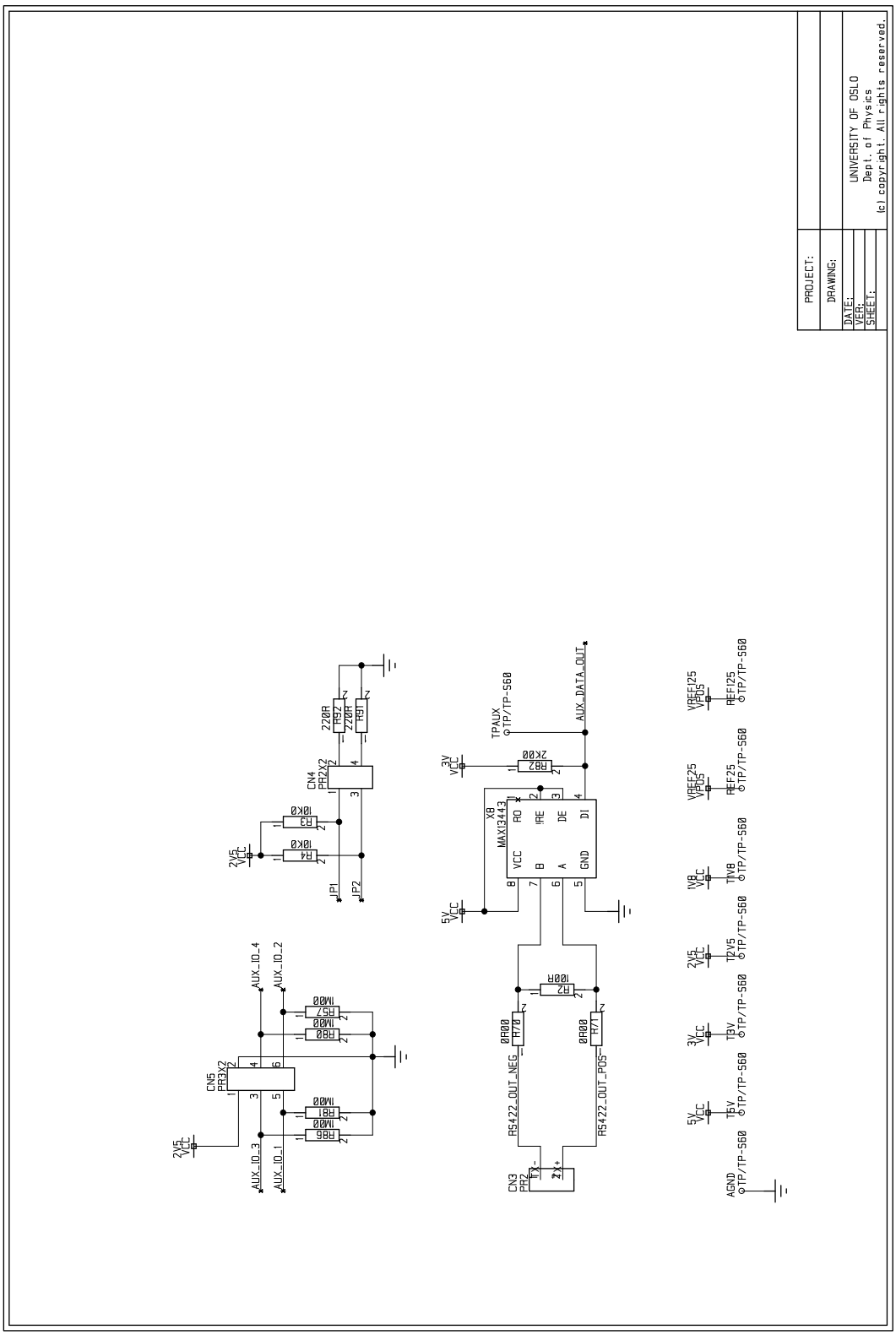

Figure D.6: Auxiliary Input/Output Block

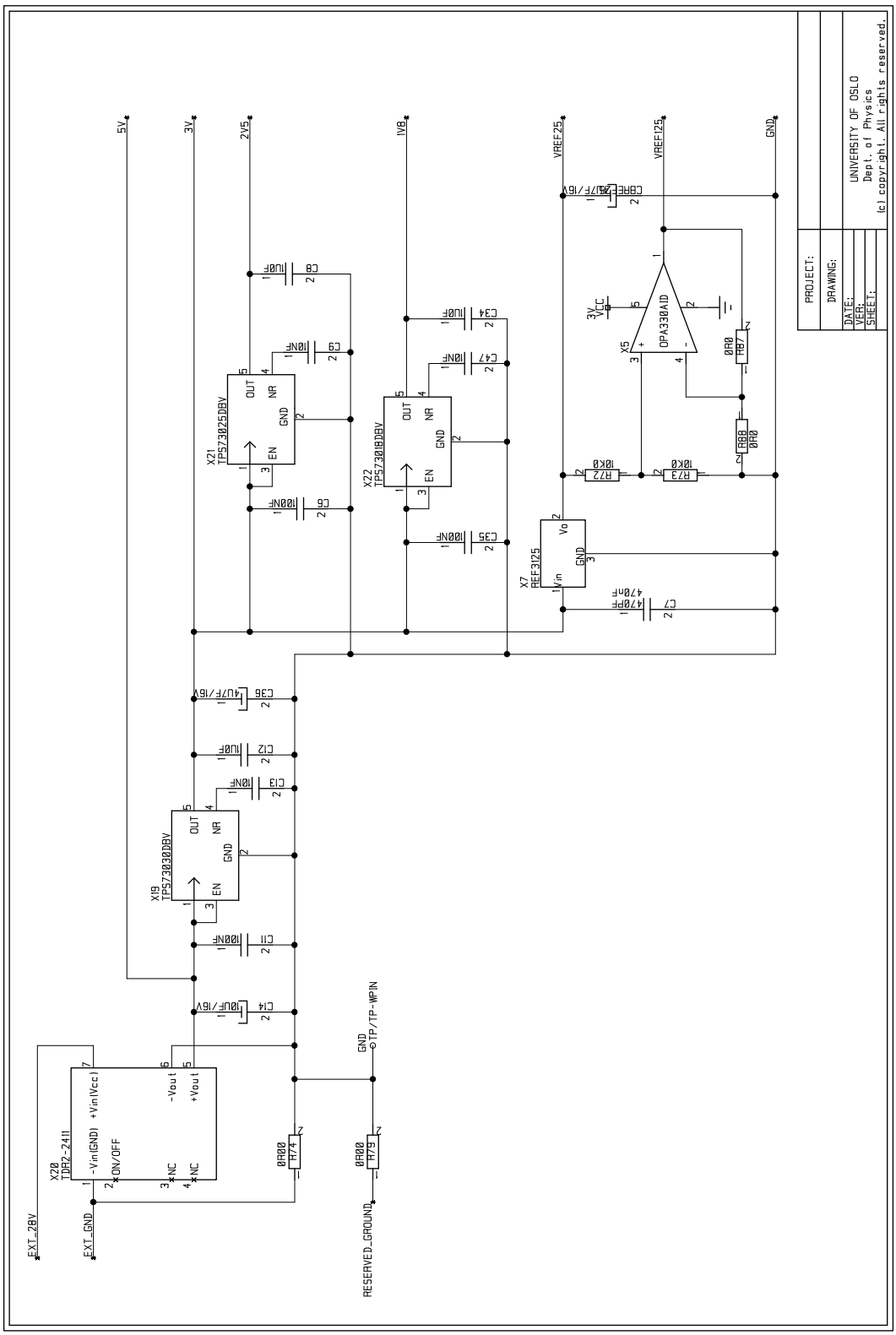

Figure D.7: Power Block

Appendix E

PCB layout v1.1

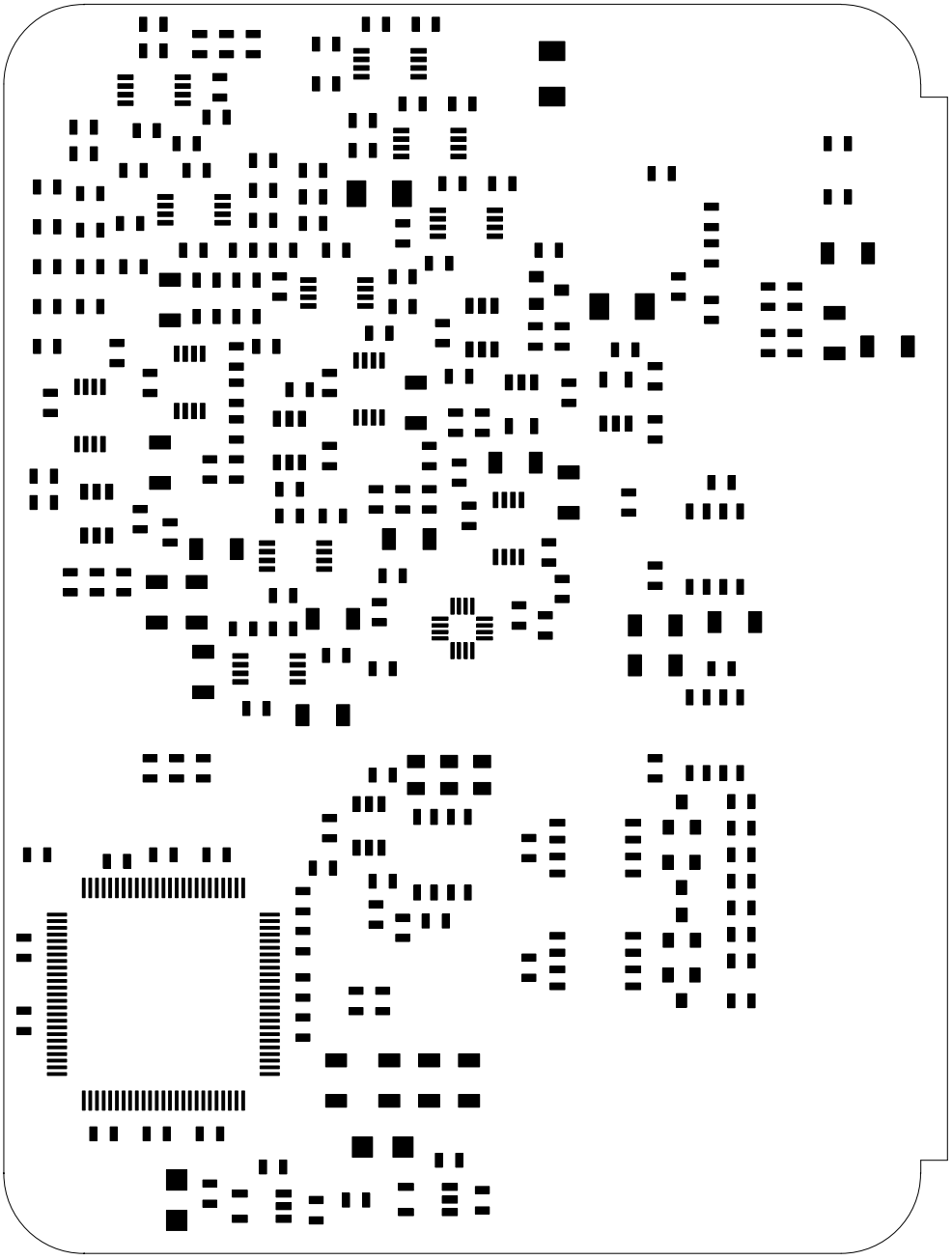

Figure E.1: Top Solder Paste

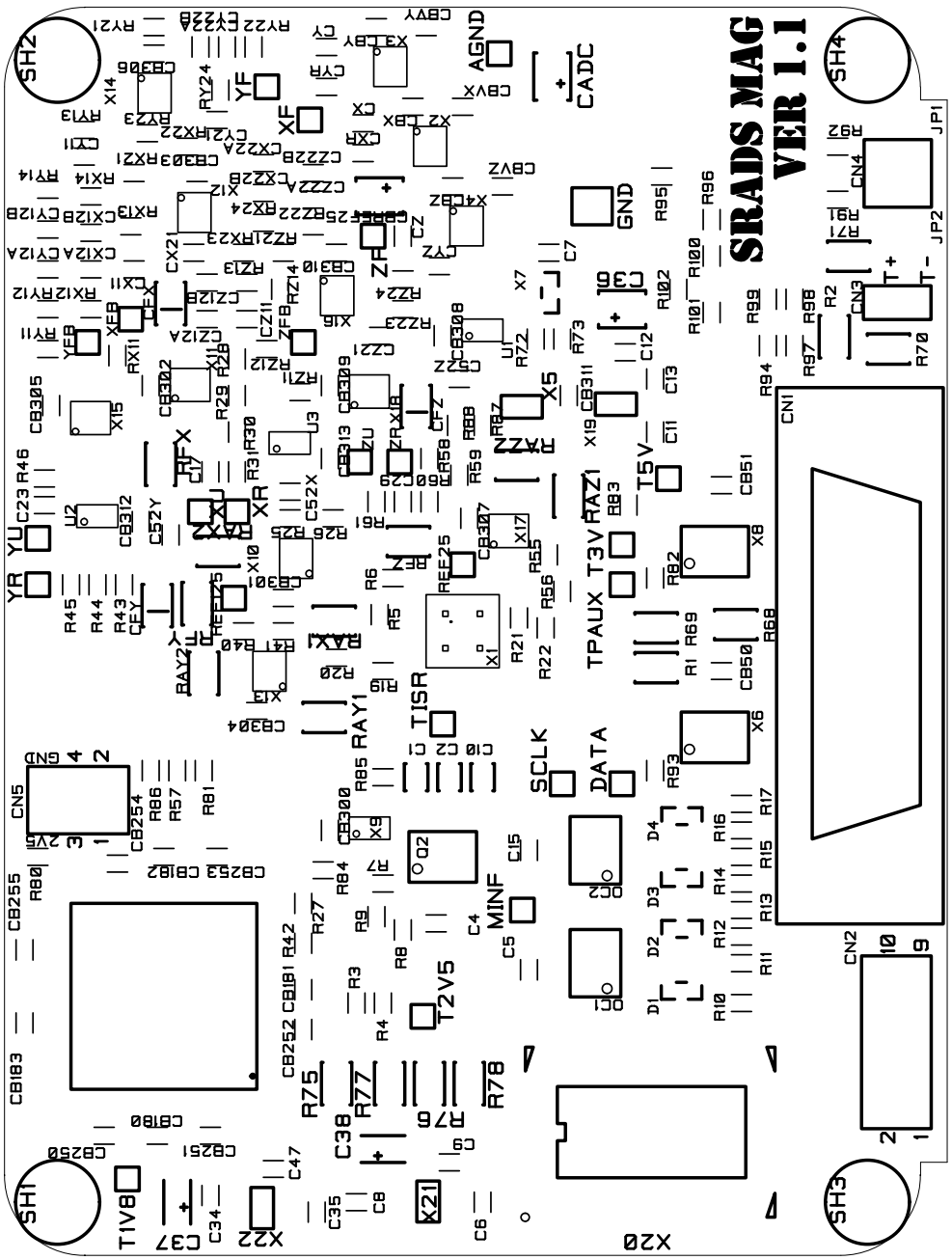

Figure E.2: Top Silk Print

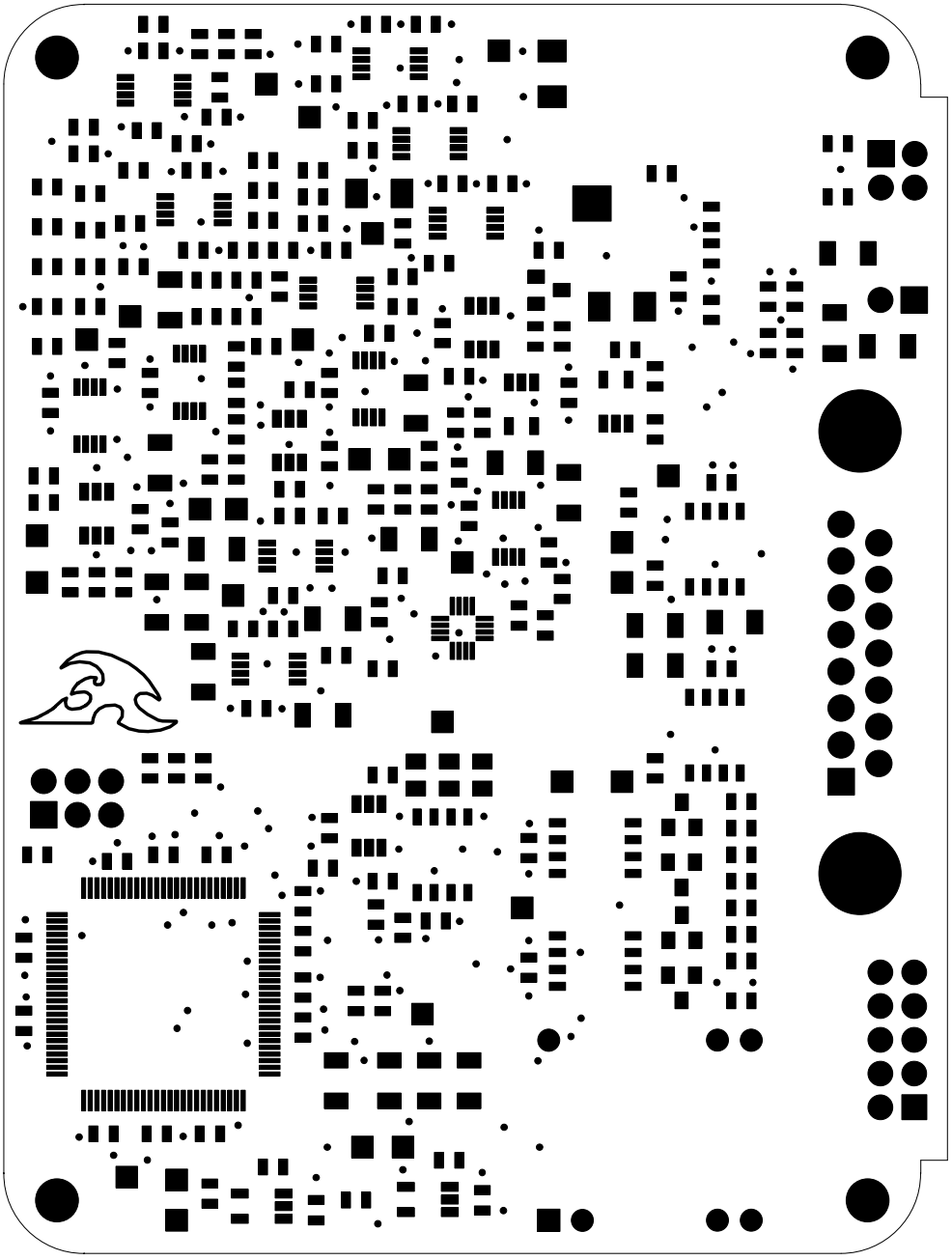

Figure E.3: Top Solder Stop

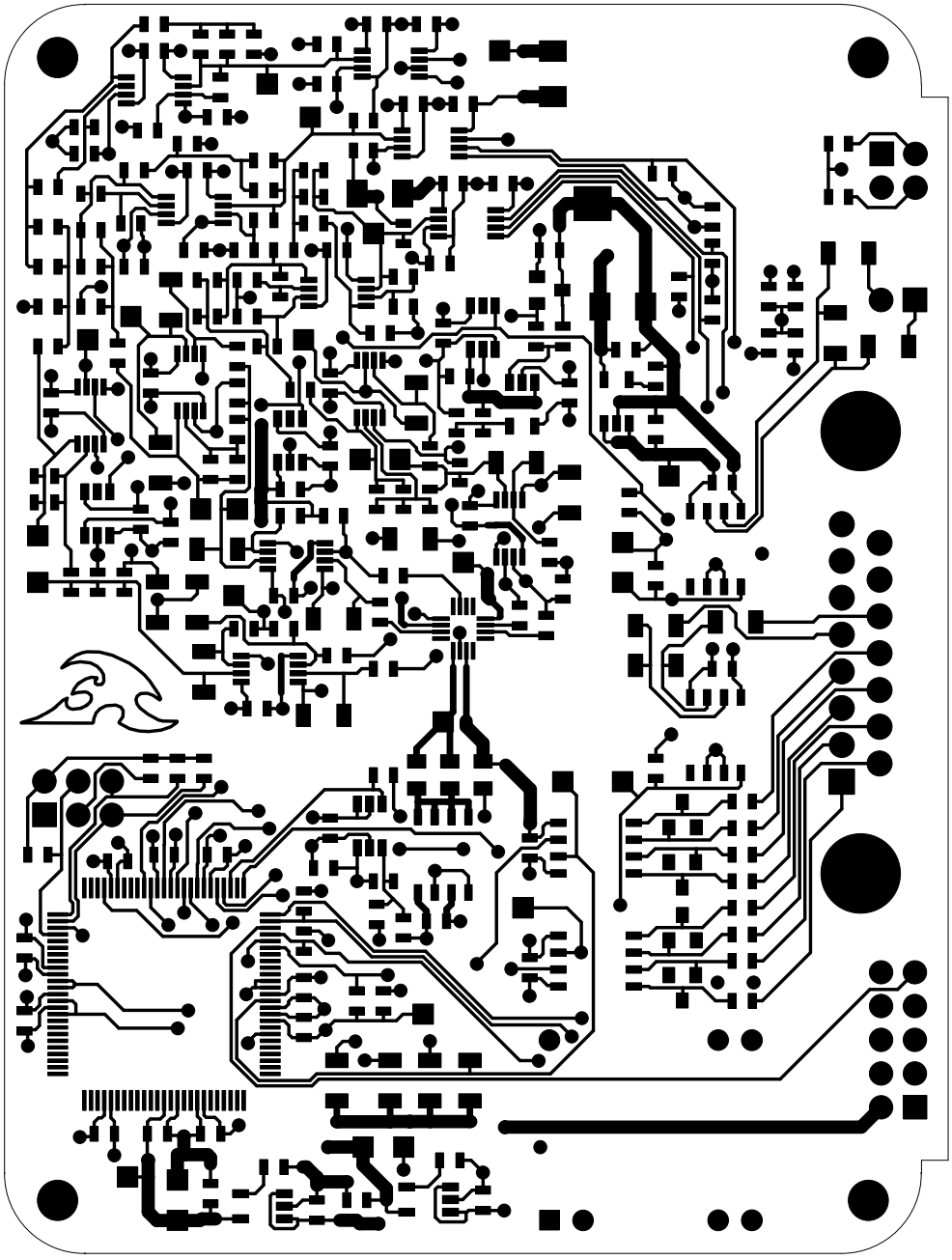

Figure E.4: Top Electric

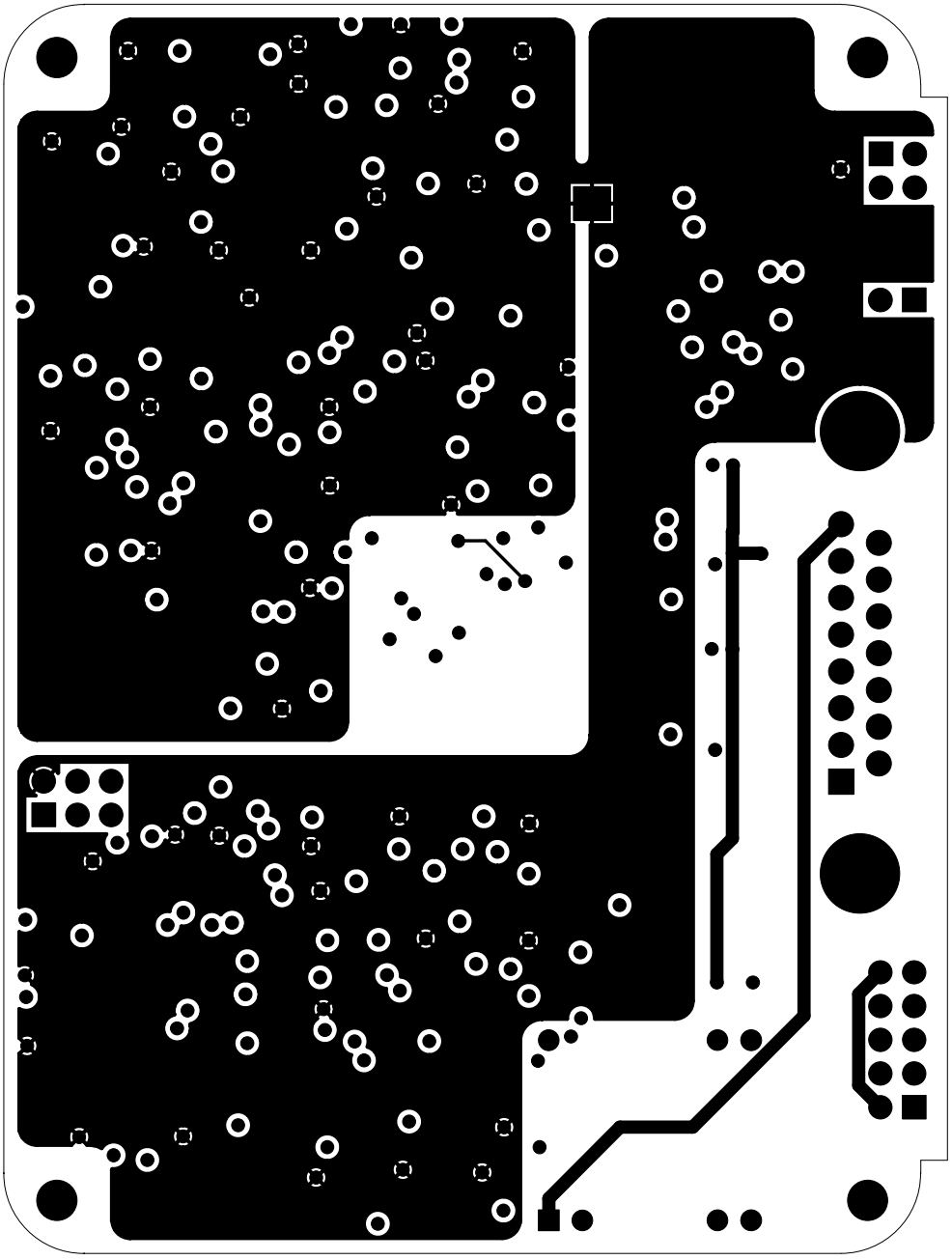

Figure E.5: Inner GND Layer

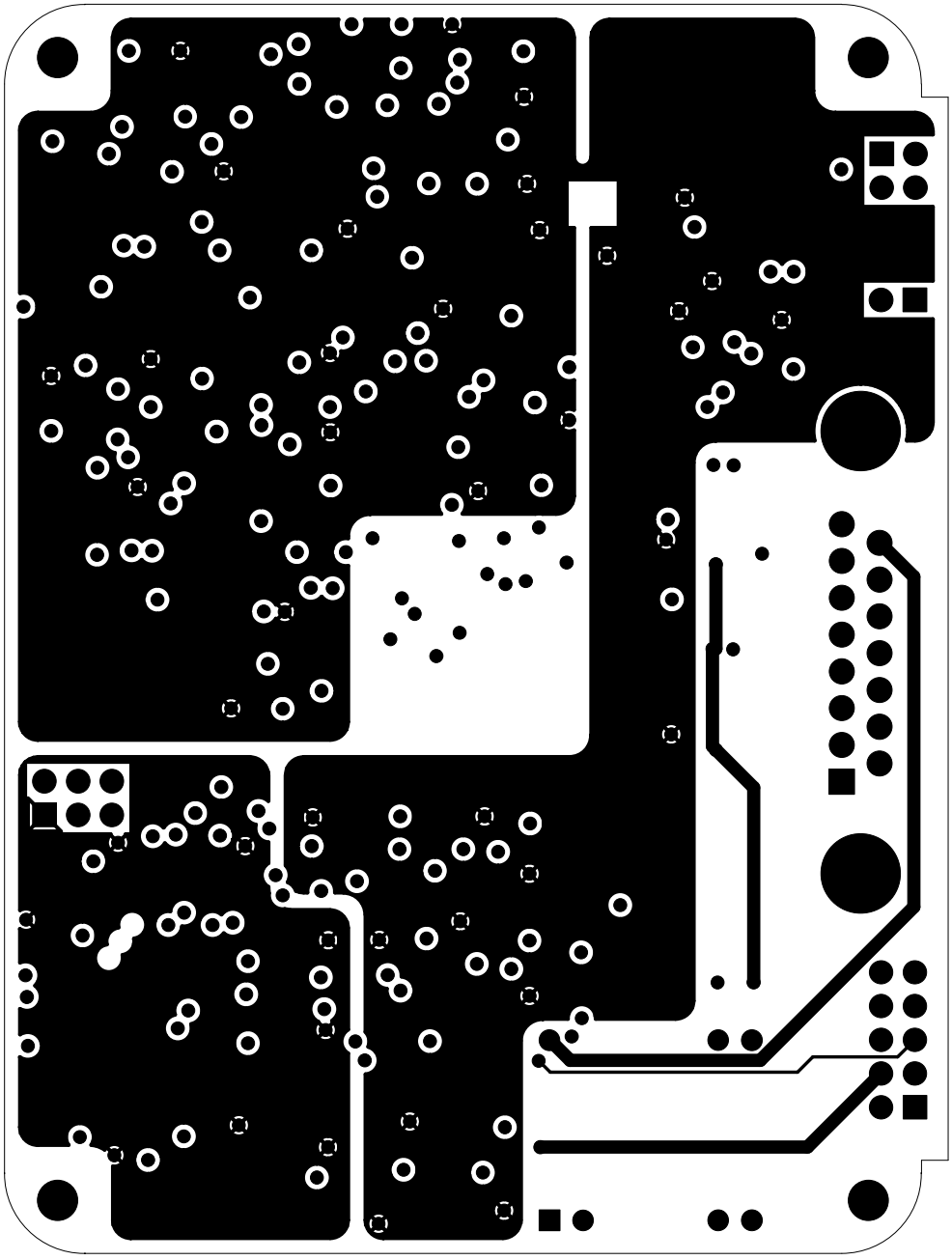

Figure E.6: Inner VCC Layer

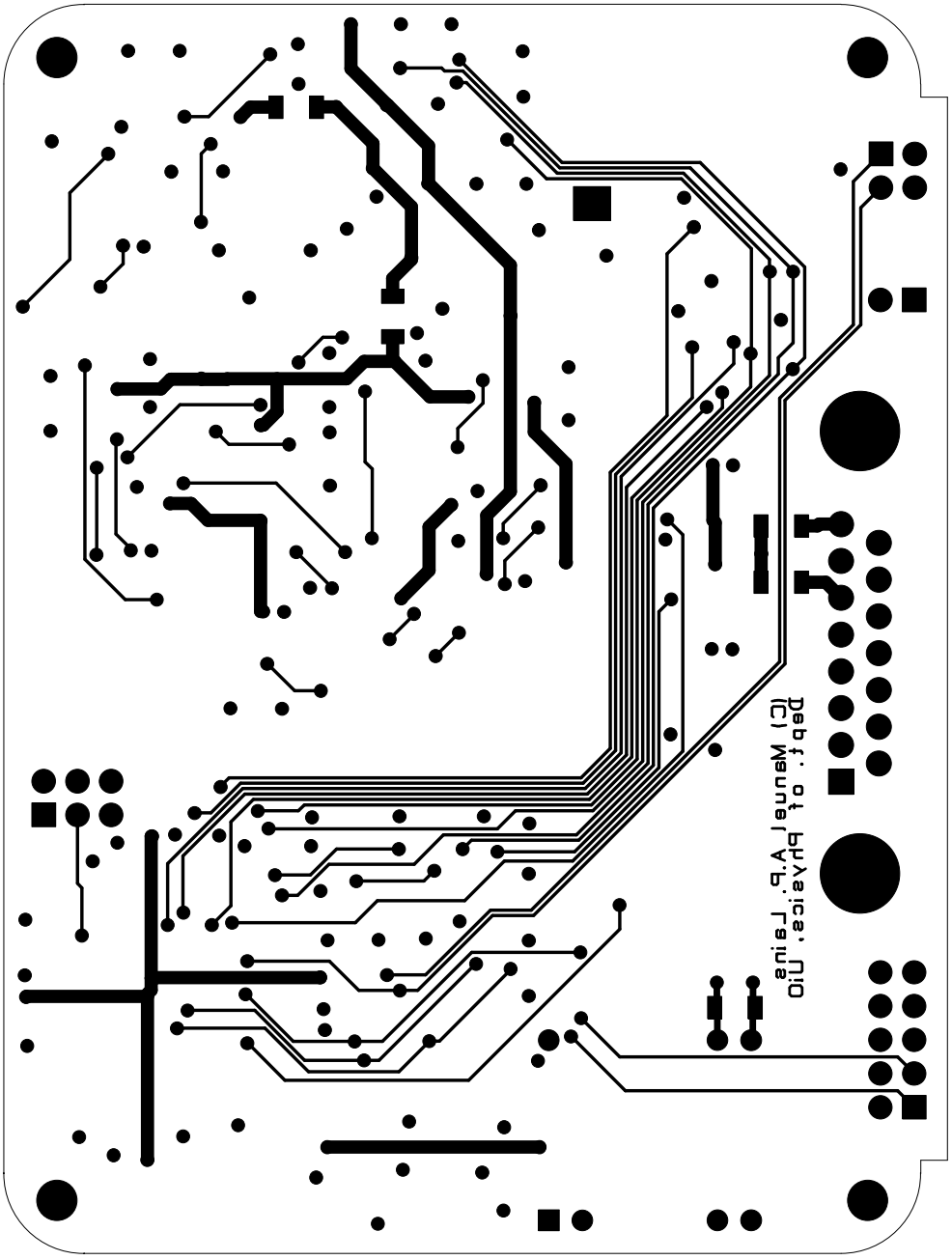

Figure E.7: Bottom Electric

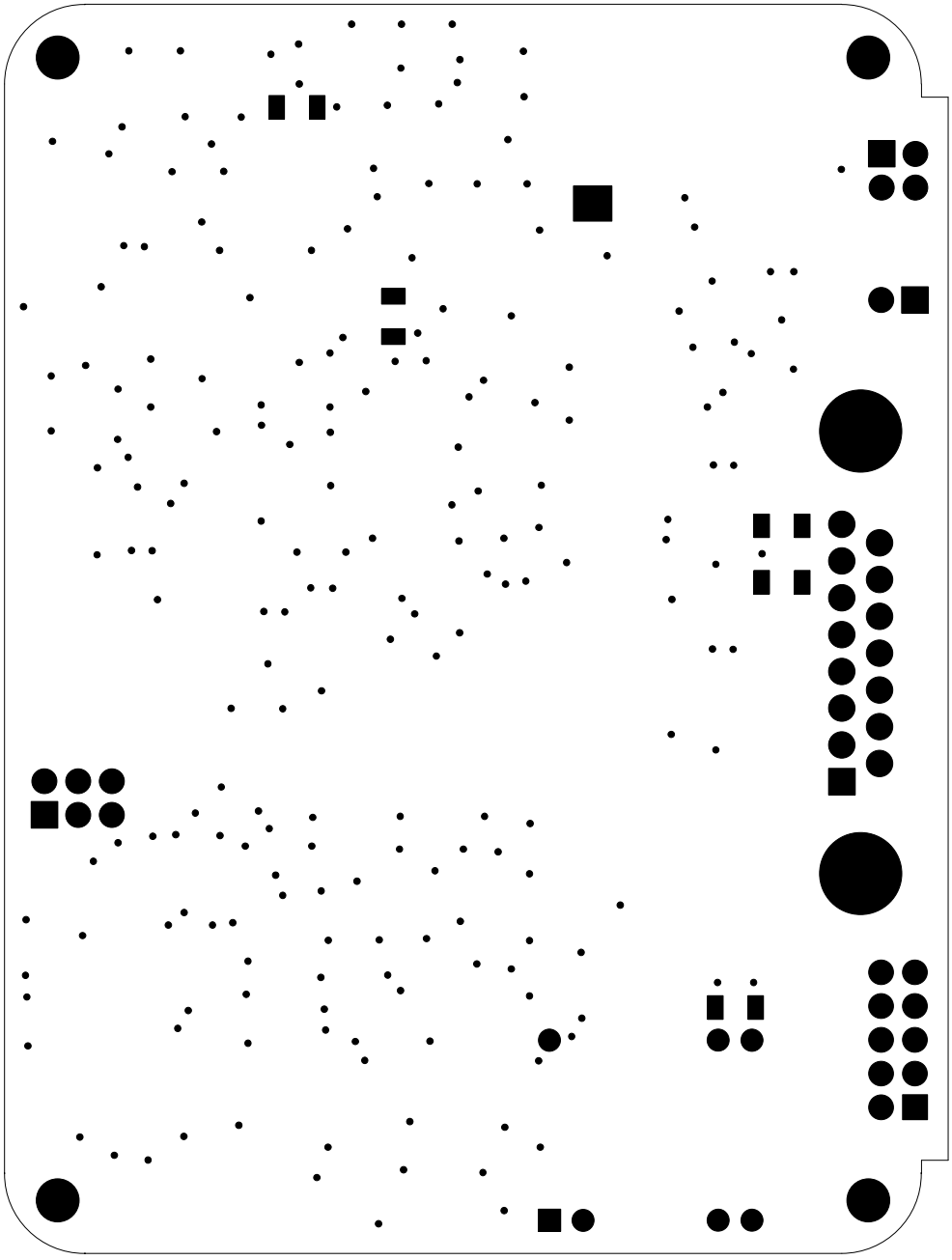

Figure E.8: Bottom Solder Resist

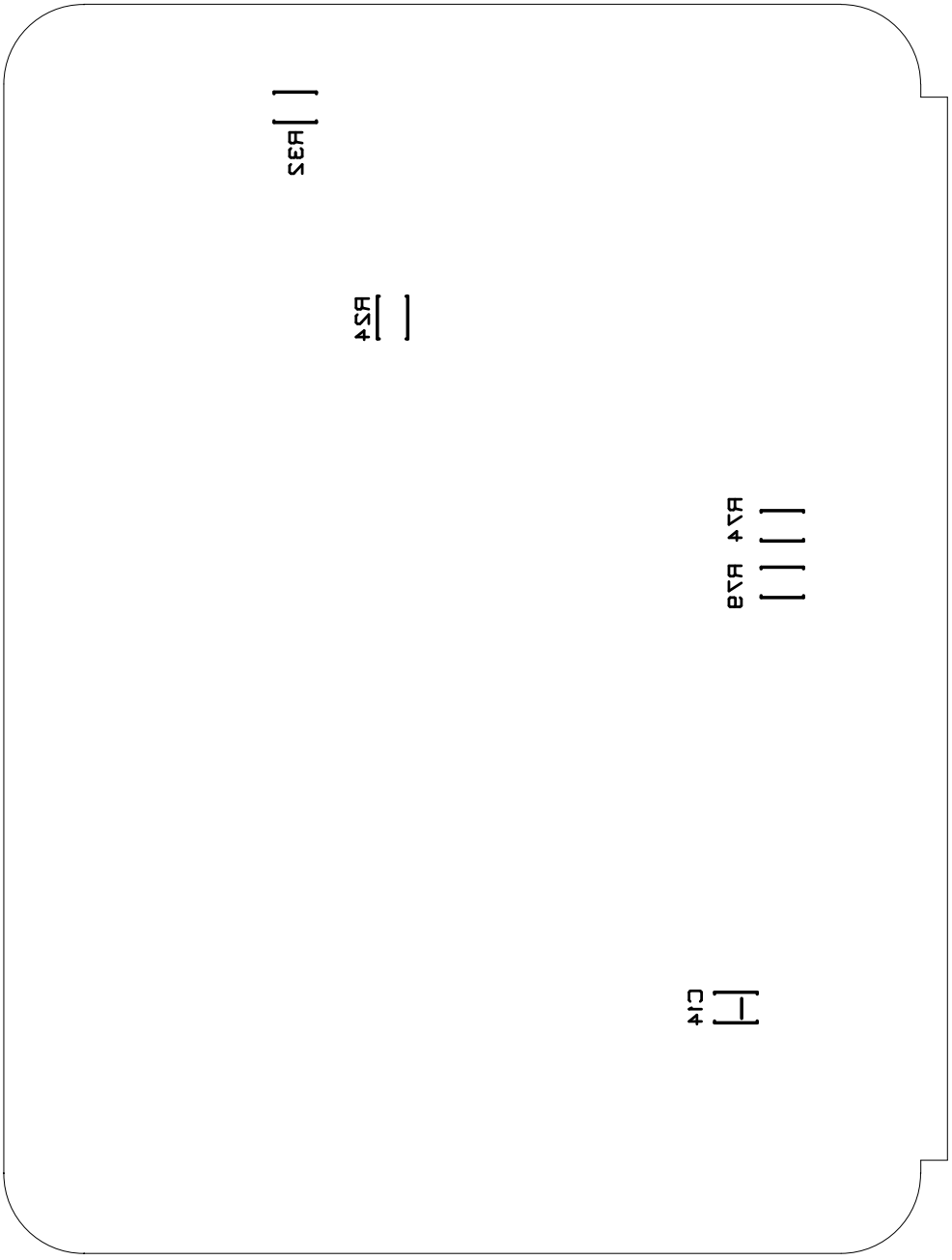

Figure E.9: Bottom Silk Print

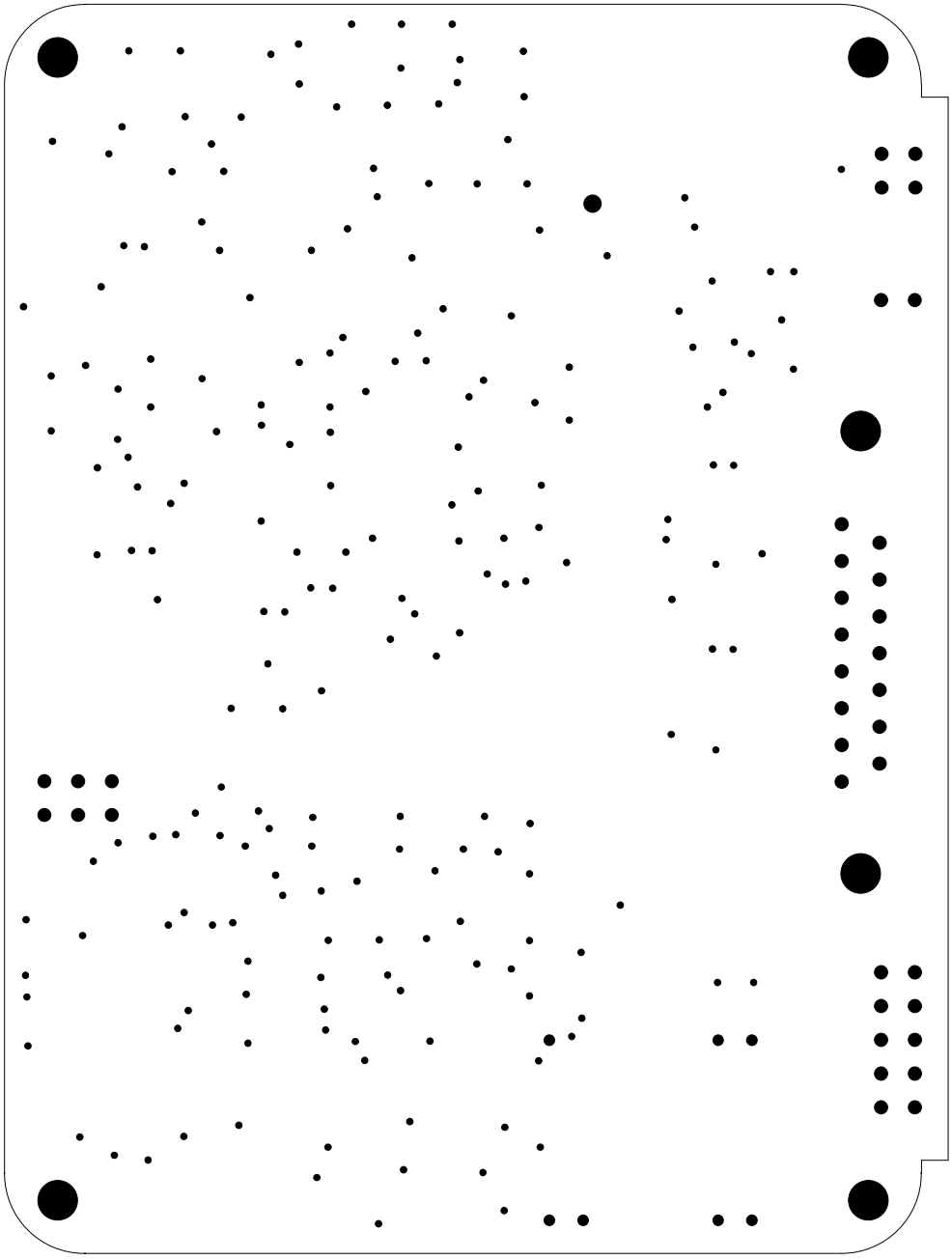

Figure E.10: Drill Holes